Reactive Deliberation: An Architecture for Real-time Intelligent Control in Dynamic Environments

(Appeared in AAAI-94, p. 1303-1308)

Michael K. Sahota
Laboratory for Computational Intelligence
Department of Computer Science
University of British Columbia
Vancouver, B.C., Canada, V6T 1Z4
sahota@cs.ubc.ca

Abstract

Reactive deliberation is a novel robot architecture that has been designed to overcome some of the problems posed by dynamic robot environments. It is argued that the problem of action selection in nontrivial domains cannot be intelligently resolved without attention to detailed planning. Experimental evidence is provided that the goals and actions of a robot must be evaluated at a rate commensurate with changes in the environment. The goal-oriented behaviours of reactive deliberation are a useful abstraction that allow sharing of scarce computational resources and effective goal-arbitration through inter-behaviour bidding. The effectiveness of reactive deliberation has been demonstrated through a tournament of one-on-one soccer games between real-world robots. Soccer is a dynamic environment; the locations of the ball and the robots are constantly changing. The results suggest that the architectural elements in reactive deliberation are sufficient for real-time intelligent control in dynamic environments.

Introduction

A robot operating within the real-time constraints of the external environment must answer the question: "What to do now?" It is not sufficient for a robot to react and interact with its environment; it must act in goal-oriented ways to produce externally observable intelligent behaviour (Brooks, 1991) and not just any behaviour. The importance of real-time control is identified by the following quote: "An oncoming truck waits for no theorem prover." (Gat, 1992) The moral is that robots operating in dynamic domains must keep pace with changes in the environment. This point has been argued more formally by Maes (Maes, 1990).

Robot architectures specify the organizing principles of a robot controller. Key issues are: the computational model used, locus of control, response time, and action selection mechanism. Depending on trade-offs made in design, architectures may only be appropriate for specific classes of problem domains. This paper argues that the challenges posed for robots in complex dynamic domains have not been adequately addressed by extant architectures and describes one possible solution.

Related Work

The Good Old Fashioned AI and Robotics (GOFAIR) (Haugeland, 1985; Mackworth, 1993) research paradigm has shaped the area of robotics since the time of the robot Shakey (Nilsson, 1984). Some of the fundamental assumptions made of the world in the pure form of GOFAIR were that there is only one agent, that the environment is static, that actions are discrete and are carried out sequentially, and that the world can be accurately and exhaustively modeled by the robot. Under these assumptions, the problem of robot control is reduced to generating a plan (a sequence of actions that will, if executed, achieve a goal) and monitoring the execution of the plan. These assumptions are invalid in complex dynamic environments where it is no longer possible to accurately predict the outcome of a sequence of actions. More recent planning-based architectures (Firby, 1992; Gat, 1992) allow for local adaptation to changes in the environment, but still commit the robot to the nearly blind pursuit of arbitrary length plans. ATLANTIS (Gat, 1992) is a notable exception since it allows the consideration of alternate plans, but the commitment to the plans-as-communication view (Agre & Chapman, 1990) prevents specific plan details from being computed until they are needed, thus resulting in a greater latency in response time.

The failure of GOFAIR has led to the development of architectures that provide a direct coupling of perception to action in order to provide highly reactive behaviour. The most notable of these is the Subsumption architecture (Brooks, 1986), where the control system of a robot is composed of a hierarchy of task-achieving behaviours in which higher levels of behaviour can subsume lower levels. The concrete-situated approach (Agre & Chapman, 1987;

Chapman, 1991) formulates the control system for a robot as a collection of action proposing modules. Conflicts between proposals for external actions are resolved through a fixed priority scheme. In the situated automata approach (Kaelbling & Rosenschein, 1990), a fixed ranking of goal priorities and a set of goal reduction rules are compiled into a set of condition-action pairs so that an appropriate action can be selected at each time step. All of these approaches allow the robot to react immediately to changes in the environment, but are based on a fixed ranking of actions (or equivalently behaviours or goals). With a fixed ranking, the designer of a robot is limited in adapting the controller to the environment. A key feature of these approaches is the ability to compile the specification for a robot controller into circuits or augmented finite state machines for fast execution. A potential drawback is that controllers based on the concrete-situated and situated automata approaches cannot perform the search-type algorithms needed for planning. Although the subsumption architecture supports arbitrary computations, the subsumption mechanism and the commitment to avoid representations seems to be a significant hinderance in the development of more sophisticated robots. Some evidence for this point is given with the discussion of experimental results.

Maes proposed action selection mechanism for dynamic domains is a network consisting of goals, input predicates, and competence modules that represent actions (Maes, 1990). Activation energy flows about the network according to the dependencies and conflicts among the elements. Global parameters allow the network to be tuned to an environment; these can be learned automatically (Maes, 1991). Possible drawbacks of this mechanism are that inputs are restricted to predicates and all goals are of equal weight. The use of predicates forces potentially useful information about the environment to be discarded, while the equal weighting of goals does not reflect the likely possibility that some goals are more important than others.

Overview

The bulk of this paper is divided into two sections. The first introduces the *reactive deliberation* architecture while the second describes the experiments used to test it. The architectural elements and the motivations for reactive deliberation — a robot architecture targeted towards dynamic domains — are discussed. A tournament of one-on-one soccer games has been conducted using real-world robots to demonstrate the utility of the proposed architecture. The use of soccer is motivated, the experimental testbed is briefly described, and the results are discussed. This paper ends with conclusions and future work.

The Reactive Deliberation Architecture

Reactive deliberation is a robot architecture that integrates reactive and goal-directed activity. Even deliberation must be to some extent reactive to respond to changes in the environment. Although the name is apparently an oxymoron, it is consistent with Artificial Intelligence nomenclature (cf Reactive Planning).

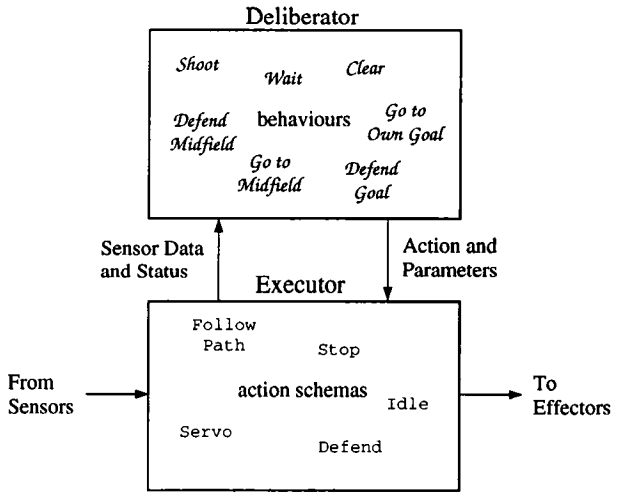

Figure 1 The Reactive Deliberation Controller

Under reactive deliberation, the robot controller is partitioned into a deliberator and an executor; the distinction is primarily based on the different time scales of interaction. Informally, the deliberator decides what to do and how to do it, while the executor interacts with the environment in real-time. These components run asynchronously to allow the executor to interact continuously with the world and the deliberator to perform time consuming computations. This partition is inspired by recent architectures that attempt to integrate planners with more reactive components (Firby, 1992; Gat, 1992) A structural model illustrating the partition with examples of a soccer-playing robot can be seen in Figure 1.

The Executor

The executor is composed of a collection of action schemas. An *action schema* is a robot program that interacts with the environment in real-time to accomplish *specific* actions. Action schemas exhibit the same level of complexity as controller modules in RAP (Firby, 1992) and primitive actions in ATLANTIS (Gat, 1992). They are designed in the spirit of behaviour-based approaches, where each schema is experimentally verified. All the schemas together define the capabilities of the robot and are independent of the robot's goals.

The deliberator enables a single action schema with a set of run-time parameters that fully defines the activity. Only one action schema is enabled at a time and it interacts with the environment through a tight feedback loop. In the world of real-time control there is no room for time consuming

planning algorithms. Computations in action schemas are restricted to those that can keep pace with the environment, so lengthy computations are performed in the deliberator.

Several examples of action schemas applicable to the soccer domain are shown in Figure 1. The *follow path* schema follows a path that consists of circular arcs and straight line segments to within a certain tolerance measured in absolute position and heading errors. The *servo* schema tries to servo the robot into the ball by driving to the predicted future location of the ball that is computed using an internal model of the ball's dynamics. The *defend* schema alternates between two modes. Normally, the robot stays between the ball and the center of the net. However, if the projected motion of the ball will carry it past the line the robot is defending, the robot moves to intercept it in an effort to keep the ball away from the net.

The Deliberator

The focus of the deliberator is on an effective mechanism for selecting actions or goals in a timely manner. A central feature of reactive deliberation is that the deliberator is composed of concurrently active modules called *behaviours* that represent the goals of the robot. The notion of a behaviour is used in the sense of Minsky's mental proto-specialists (Minsky, 1986). The examples given in Figure 1 illustrate the goals of a simple soccer-playing robot. These include goals of achievement such as *shoot* or *clear* the ball and goals of prevention such as *Defend Goal* where goals are prevented from being scored by the other robot.

A *behaviour* is a robot program that computes an action that may, if executed, bring about a specific goal. Behaviours propose actions whereas action schemas perform actions. Each behaviour must perform the following: 1) select an action schema, 2) compute run-time parameters for the schema (plan the action), and 3) generate a bid describing how appropriate the action is. The most appropriate behaviour, and hence action, is determined in a distributed manner through inter-behaviour bidding.

Each bid is an estimate of the expected utility and is based on the current state of the world as well as the results of planning. Currently, the criteria for generating the bids are hand coded and tuned so that the most appropriate behaviour is active in each situation. This approach requires the designer of a system to explicitly state the conditions under which certain behaviours are suitable or favourable. A simplified version of this appears in architectures with fixed ranking schemes. For example, the concrete-situated approach uses binary preference relations to establish an ordering of proposers or actions.

Modularity The principal advantage of behaviour-based bidding is modularity. Since bids are calibrated to an external measure of utility, behaviours can be added, modified or deleted without changing the bidding criteria of the established system. A new behaviour must, of course, be tuned to be compatible with existing ones. Behaviours are independent, so they can have different representations and approaches to generating actions. For instance, a behaviour could incorporate a traditional planner and generate a bid that reflects the utility of the current step of the plan. There is no central decision maker that evaluates the world and decides the best course of action, so behaviours can be run concurrently on different processors (instead of timesharing a single processor), thus improving the speed of the system. In our approach, there is no negotiation between behaviours, unlike in systems such as contract nets (Smith, 1980). As a result, it is not possible to combine the preferences of multiple behaviours, and this remains an open problem.

Real-time computations In a real robot there is more to the problem of action selection than just deciding what to do. In dynamic environments, a robot needs to quickly decide what to do and how to do it. The deliberator must keep pace with changes in the environment to produce intelligent behaviour. Each behaviour is responsible for computing a bid and planning the action. Fixed computational resources (processor cycles) need to be distributed among the behaviours, since it is typically the case that there is too much computation to be done.

The exact mechanism for distributing computational resources is left unspecified as it is strongly dependent on the real-time requirements of the system, the number of behaviours, and the resources needed by each behaviour. However, the basic principle is to divide the available computational resources among the behaviours such that the ruling behaviour receives more resources. This allows behaviours that perform minimal computations to respond quickly, while those that perform lengthy computations will respond slowly. It might be appropriate to allocate resources according to the importance and needs of each behaviour, but there are no provisions for this in the current implementation. There is no perfect architectural solution to the problem of limited computational resources: if the computations are slow, then the robot will be slow too. The only possible solutions are to get more computers, faster computers, better algorithms, or switch to simpler tasks.

Why this partition?

Reactive deliberation, like GOFAIR approaches, partitions the controller for a robot into a deliberator and executor. One difference is the level of abstraction at which the split between reasoning and execution monitoring occurs; our claim is that the reactive deliberation split is more suitable for dynamic environments.

The deliberator is responsible for answering the questions: "What to do now?" and "How should it be done?" Believers in the theory of plans-as-communication (Agre &

Chapman, 1990) argue that these questions can and should be resolved independently (Gat, 1992). In this case a planner decides what to do based on an abstract world model, while the problem of resolving how each action should be performed is postponed until it is to be executed. In a dynamic environment, however, these questions are usually interrelated. Before committing to an action, it is important to verify that the action is both feasible and more appropriate than other actions. Architectures that follow the planning paradigm check to see if an action is feasible, but not if there is a better action.

Answering the question "How should it be done?" provides information about the utility of an action. For example, detailed planning may show that one action is impossible, while another can be accomplished quickly. This suggests that generating plans at a high level of abstraction may not provide an effective solution for the problem of action selection. Unless all actions of the robot are feasible and the outcomes can be predicted at design time, the question "What to do now?" cannot be intelligently answered without also answering "How should it be done?"

Another advantage of reactive deliberation is that the deliberator is responsible for generating a single action (schema), whereas other planning-based architectures generate a complete plan (i.e. sequences of actions). This distinction allows behaviours to focus on either the immediate situation or some interval of time depending on what is appropriate.

The boundary between appropriate and inappropriate computations in the executor is a function of the computing power of a particular system and specific environmental constraints. Any computations that can be performed within the time constraints of the environment are suitable for use in the executor. All other computations are relegated to the deliberator to avoid degrading the ability of the robot to interact in real-time. Regardless of advances in computing power, there will likely be interesting algorithms that do not run in real-time. This suggests that the partition between the executor and the deliberator is indicative of a technology-independent need to partition computations.

Soccer-playing Experiments

This section links theory to practice through a robot controller that has been constructed using reactive deliberation. The controller has been designed so that the robot can compete with another robot in a one-on-one game of soccer. In this section, the use of soccer is motivated as an appropriate domain for robotic experiments in dynamic domains. The testbed used to perform the experiments is briefly described. The experimental results are presented and their implications for robot architectures are discussed.

Why soccer?

Soccer has characteristics prevalent in the real-world that are absent from typical robot problem domains (Sahota & Mackworth, 1994). Soccer-playing is a dynamic environment because the ball and the cars are all moving. A robot must deal with cooperating agents on the robot's team, competing agents on the other team, and neutral agents such as the referee and the weather. The world is not completely predictable: it is not possible to predict precisely where the ball will go when it is kicked, even if all the relevant factors are known. Continuous events such as a player running to a position and the ball moving through the air occur concurrently add further complexity.

One advantage of the soccer domain is that there are objective performance criteria; the ability to score and prevent goals and the overall score of the game allow explicit comparisons of alternative controller designs. The ability to compare controller designs and draw conclusions from their strengths and weaknesses is a central feature of this domain.

One problem with a direct comparison of robot controllers is that differences in performance may be the result of technical details (such as the length of time the designer spent tuning the controller) that may have nothing to do with the underlying architectures. However, implemented systems can provide a lower bound on the utility of an architecture since limitations in the architecture are often reflected in the functionality of a robot. For the experiments described in this paper, the problem is avoided by using the same program fragments in each controller with different organizational principles.

The Dynamite Testbed

A facility called the Dynamite testbed has been designed to provide a practical platform for testing theories in the soccer domain using multiple mobile robots (Barman *et al.*, 1993). It consists of a fleet of radio controlled vehicles that perceive the world through a shared perceptual system. In an integrated environment with dataflow and MIMD computers, vision programs can monitor the position and orientation of each robot while planning and control programs can generate and send out motor commands at 60 Hz. This approach allows umbilical-free behaviour and very rapid, lightweight fully autonomous robots.

The mobile robot bases are commercially available radio controlled vehicles. We have two controllable 1/24 scale racing-cars, each 22 cm long, 8 cm wide, and 4 cm high excluding the antenna. The testbed (244 cm by 122 cm in size) with two cars and a ball is shown in Figure 2. The cars have each been fitted with two circular colour markers allowing the vision system to identify their position and orientation. The ball is the small object between the cars.

A feature of the Dynamite testbed is that it is based on the

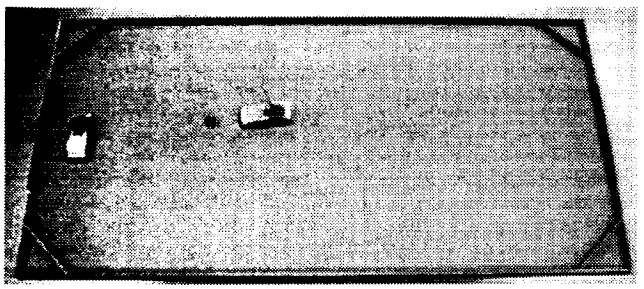

Figure 2 Robot Players on the Soccer Field

"remote brain" approach to robotics. The testbed avoids the technical complexity of configuring and updating on-board hardware and makes fundamental problems in robotics and artificial intelligence more accessible. We have elected not to get on-board the on-board computation bandwagon, since the remote (but untethered) brain approach allows us to focus on scientific research without devoting resources to engineering compact electronics.

A physics-based graphics simulator for the Dynamite world has been used for testing and developing reasoning and control programs.

Results

Several controllers based on reactive deliberation have been implemented to allow robots to compete in one-on-one games of soccer. Current functionality includes various simple offensive and defensive strategies, motion planning, ball shooting and playing goal. The robots can drive under accurate control at speeds up to 1 m/s, while simultaneously considering alternate actions. We have produced a 10 minute video that documents these features.

A series of experiments, soccer games, called the Laboratory for Computational Intelligence (LCI) Cup were performed using the Dynamite testbed (Sahota, 1993). The most elaborated reactive deliberation controller competed with subsets of itself to provide, through the scores of the games, an objective utility measure for some of the architectural features of reactive deliberation. The results of the soccer tournament that has been conducted in our laboratory can be seen in Table 1. The versions of the controller used were:

- **Reactive Deliberation**: the controller performs concurrent deliberation and execution, as is intended of the architecture.
- **Half-wit**: the executor yields control to the deliberator only when an action (activity) has been completed or a time-out occurs; this is equivalent to a GOFAIR controller.
- **No-wit**: the controller alternates between offensive and defensive behaviours according to a fixed timer regardless of the current world state.

Controller	No- wit	Half-wit	Reactive Deliberation
Reactive Deliberation	11 - 1	7 - 4	3 - 3
Half-wit	6 - 3	5 - 5	
No-wit	8 - 2		

Table 1 Final Scores in the Soccer Tournament (11 – 1 means that the reactive deliberation controller scored 11 goals while the no-wit controller scored only 1.)

There is an element of chance in these soccer games: the scores are a result of a complex set of interactions between the robots and their environment. These results are partially repeatable because the same general results will emerge, but the actual scores will be different. For a better estimate of the results, the duration of the soccer game could be extended from the current time of 10 minutes. Playing multiple games is equivalent to extending the duration of a single game.

The rank of the controllers from best to worst is: reactive deliberation, half-wit, and no-wit. This ranking is probably reliable since the better controllers scored nearly twice as many goals (7–4 and 6–3 are the scores) as the controller ranked beneath it. The results of the games played with the same controller indicate that the better two controllers (reactive deliberation, half-wit) generate fairly constant performance, while the no-wit controller produces somewhat random performance. The scores (5–5 and 3–3) should be interpreted as close scores, rather than identical. They really do not show the underlying randomness that *is* present as might be shown by a listing of when the goals were scored. The score 8–2 in the no-wit vs. no-wit game is a result of the almost random playing strategy of that controller.

The reactive deliberation controller performs better than a human controlling the opposing robot. This is, however, somewhat of an unfair comparison since excellent motor skills are needed to even shoot the ball.

Discussion

The difference in score between the reactive deliberation and half-wit controllers is significant. The only difference between these two controllers is that reactive deliberation considers alternate actions all the time, while the half-wit controller does so only when an action schema terminates. The reactive deliberation controller selects goals as frequently as possible and can interrupt actions. The half-wit controller is like the traditional planning-based architectures: alternate actions are considered only when the current action has terminated. This is evidence that the frequent evaluation of goals and actions is critical to success

in dynamic worlds.

The level of performance that the robots were able to achieve is partially due to the use of internal world models. An internal model of the dynamics of the robot is used to provide feed-forward control. This is not a superfluous element; it really is necessary for the robots to operate at speeds of 1 m/s. Brooks argues that "the world is its own best model" and that internal models are inappropriate (Brooks, 1991). Experiences with these soccer-playing robots suggest that Brooks' slogan is misleading and that either explicit or implicit models are needed.

The performance of the robots is largely a function of the action selection mechanism. It has been fine-tuned through an iteration cycle with observations of soccer games followed by incremental changes to the behaviours. A useful abstraction that helps with this is the *routines* of action from the concrete-situated approach (Agre & Chapman, 1987). The central idea is that the agent (or robot) interacts with the environment in a routine or typical way. One routine in soccer is: clear the ball, defend red line, shoot, etc. In the case of soccer-playing, the construction of successful robots does involve careful attention to patterns of activity. This is an emergent (and surprising) result of our experiments.

The reactive deliberation controller plays a nice, although not flawless, game of soccer. The competitive nature of soccer places very strict time constraints on the robots and allows different controllers to be easily compared. The dynamic and unpredictable nature of one-on-one robot soccer favours approaches that are concerned with the immediate situation and reactive deliberation takes advantage of this.

Reactive deliberation is not a panacea for robotic architectural woes. A further disclaimer is that it is an incomplete robot architecture since it focuses on the issues related to dynamic domains and ignores a number of issues such as perceptual processing and the development of world models. The proposal is orthogonal to those issues.

Conclusions

The theoretical contributions of reactive deliberation to the design philosophy of robot architecture for dynamic environments are the following:

- A new split between reasoning and control is proposed since utility and hence action selection cannot always be suitably determined without detailed planning.
- Goal-oriented *behaviours* are a useful abstraction that allow sharing of scarce computational resources and effective goal-arbitration through inter-behaviour bidding.

A series of one-on-one soccer games have been conducted with real-world robots to evaluate reactive deliberation. The score of a soccer game provides an objective criterion for evaluating the success of a robot controller. The experimental results suggest that the architectural elements in reactive deliberation are sufficient for generating real-time intelligent control in dynamic environments. Further, it has been experimentally demonstrated that the goals and actions of a robot need to be evaluated at a rate commensurate with changes in the environment.

Future Work

Future work can be classified as either testing or extensions. Possible testing procedures include comparing reactive deliberation with other architectures and testing it in other problem domains. It is not clear how general reactive deliberation is and it remains to be determined in which domains this style of architecture is preferable. One limitation of this research is that the experiment, from proposed solutions to testing, has been performed by the author; a more hands-off or double-blind procedure is needed to provide greater scientific rigour. It still remains to be demonstrated that our architecture is more appropriate than others even in the particular soccer-world that has been used.

Some possible extensions to reactive deliberation are as follows:

- incorporation of perceptual processing and world modeling into the architecture.
- development of a more formal, yet practical, mechanism for estimating utility.
- learning utility estimates and models of the robot's dynamics.
- ability to combine the preferences of different behaviours.
- support for inter-robot cooperation.

Acknowledgments

I am grateful to Rod Barman, Keiji Kanazawa, Stewart Kingdon, Jim Little, Alan Mackworth, Dinesh Pai, Heath Wilkinson and Ying Zhang for help with this. In particular, Alan has provided deep insights and help revising drafts. This work is supported, in part, by the Canadian Institute for Advanced Research, the Natural Sciences and Engineering Research Council of Canada and the Institute for Robotics and Intelligent Systems Network of Centres of Excellence.

References

Agre, P., and Chapman, D. 1987. Pengi: An implementation of a theory of activity. In *AAAI-87*, 268–272.

Agre, P., and Chapman, D. 1990. What are plans for? In Maes, P., ed., *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back.* M.I.T. Press. 17–34.

Barman, R.; Kingdon, S.; Little, J.; Mackworth, A. K.; Pai, D.; Sahota, M.; Wilkinson, H.; and Zhang, Y. 1993. Dy-

namo: real-time experiments with multiple mobile robots. In *Proceedings of Intelligent Vehicles Symposium*, 261–266.

Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* RA-2:14–23.

Brooks, R. A. 1991. Intelligence without reason. In *IJCAI-91*, 569–595.

Chapman, D. 1991. *Vision, Instruction, and Action.* MIT Press.

Firby, R. J. 1992. Building symbolic primitives with continuous control routines. In *First International Conference on Artificial Intelligence Planning Systems*, 62–69.

Gat, E. 1992. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *AAAI-92*, 809–815.

Haugeland, J. 1985. *Artificial Intelligence: The Very Idea.* Cambridge, MA: MIT Press.

Kaelbling, L. P., and Rosenschein, S. J. 1990. Action and planning in embedded agents. In Maes, P., ed., *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back.* M.I.T. Press. 35–48.

Mackworth, A. 1993. On seeing robots. In Basu, A., and Li, X., eds., *Computer Vision: Systems, Theory, and Applications*. World Scientific Press. 1–13.

Maes, P. 1990. Situated agents can have goals. In Maes, P., ed., *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back.* M.I.T. Press. 49–70.

Maes, P. 1991. Learning behaviour networks from experience. In *Proceedings of the First European Conference on Artificial Life*. M.I.T. Press.

Minsky, M. 1986. *The Society of Mind.* Simon & Schuster Inc.

Nilsson, N. 1984. Shakey the robot. Technical Report 323, SRI International. Collection of Earlier Technical Reports.

Sahota, M. K., and Mackworth, A. K. 1994. Can situated robots play soccer? In *Proceedings of Canadian AI-94*, 249–254.

Sahota, M. K. 1993. Real-time intelligent behaviour in dynamic environments: Soccer-playing robots. Master's thesis, University of British Columbia.

Smith, G. 1980. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computing* 29(12).

Responses to Symposium Questions

For many of the questions posed here, there is no universal answer. Instead, there is often a trade-off that depends on the particular robot that is used or task that is performed. For instance, a large outdoor robot with an extensive vision system *should* be designed in radically different ways than a desktop robot with four infrared sensors. One purpose of these questions may be to elucidate what the trade-offs are in order to reach some conclusions about what types of architectures make sense for a robot of type X performing task Y in environment Z.

A deeper interpretation of the questions is how can we build more sophisticated robots than the current ones? I think that this an important scientific goal, but I don't think that robot architectures can provide any answers since the problem lies elsewhere. The current bottlenecks are sensor processing and providing sensor-motor control. An informal thesis I have is, "Once you have written the perceptual system and the low-level control, you can write the decision-making system on the back of an envelope (given the limitations of today's robots and the typical tasks they are set to). Sophisticated architectures are only useful for robots performing sophisticated tasks. Admittedly robot architecture may have a contribution to make in this, but so does software engineering.

Many robots are limited in their ability to interact with the world due to primitive perception systems. I think that significant advances in robots will follow advances in computer vision. This may take a while since many computer vision researchers are more interested in elegant mathematics than in working systems.

Coordination

A wide range of models, from symbolic systems to subsumption, have been proposed for sequencing behaviours or actions. I think that the appropriateness of a particular model depends on the task of the robot. For robot soccer (a competitive dynamic environment), one conclusion we have reached is that alternative actions must be considered at a rate commensurate with changes in the environment.

The problem of arbitrating among goals and composing constraints is a difficult one. It is useful to distinguish between two classes of constraints: hard and soft. Hard constraints are those that are incompatible with other alternatives in the action space of the agent. Soft constraints are ones that can be broken and are usually associated with some sort of cost. Soft constraints are often expressed as one of the following: energy functions, vector fields, utility, and fuzzy rules. The basic problem that arises is one of finding a suitable metric for determining the value of alternative decisions. This is usually encoded by hand in whatever arbitration mechanism makes the most sense for the robot's action space and task.

Representation

There is an interesting trade-off in the choice of representations that are used. The basic problem is that it is hard to pick a representation that is useful to both the robot and the system designer.

One extreme is to store all data in a symbolic system that

is accessible to humans. The advantage of this is that we have to design and build the system, so it is probably a good idea if it is cognitively penetrable. The problem is that it is difficult to convert from raw sensor data to conceptual terms that we like to think in (e.g. on(blockA,blockB)). Some effort in this direction appears as work on *perceptual invariants.*

The other extreme is to store all data as raw sensor values and not assign meaning to internal connections. The advantage of this is that the robot needs to make decisions based on its sensory input, so it makes some sense to compute in this space. The disadvantage is that it may not be clear why a system is or isn't working.

Structural

The human brain has different organizational properties depending on function. For instance, the part responsible for visual processing varies in spatial organization from none to very uniform (retinotopic). This suggests that it may be advantageous to use different types of computer hardware for visual processing in robots if only for reasons of computational efficiency.

Another lesson that can be drawn from the human brain is that feedback plays an important role in many functions. So we should not expect to be able to separate the functions into a unidirectional flow. The way visual inputs are processed will almost certainly depend on the goal of the robot, just as the goals of the robot depend on the visual input.

Performance

The domain we have investigated, soccer, is dynamic, uncertain, and actively hostile. The score of a game provides an objective measure of performance. This alone is not sufficient for comparing architectures - the qualitative reasons for failure are often suggestive of limitations. In any case, some performance measure is necessary if one seeks to decide if an architecture is successful or not. One serious problem with this area is that it is acceptable to say, "I designed this architecture for this robot and look - it works!"

Simulation

The motivation for using simulators is that they are often much easier to use than real robots. Real robots often have embedded computers that are difficult to debug. Real robots can damage themselves as well as people or equipment. Real robots often run out of batteries. If robots were as easy to use as a simulator and there were enough robots to go around, then no one would use a simulator. Given that robots will always tend to have the problems stated above, it is really handy to have a simulator around while you are developing code.

It is important to remember that simulations are abstractions of the world that are intended to capture the salient features of the domain the robot is operating in. Few simulators have accurate models of sensors, environment, robot plant, and dynamics since it is a non trivial problem to obtain these models. I don't foresee much use in verifying the properties of a robot against a specification (or simulator) since the it is unlikely that the specification will match the environment.

It is possible to build a *standard* simulator that various groups can use, but I don't think that it is feasible. The main problem that I foresee with this is that everyone seems to be solving different problems in different domains. Further, I don't think there is a universal architecture that is appropriate in all situations, so I don't think there can be a universal simulator.