Hierarchy and Sequence vs. Full Parallelism in Reactive Action Selection Architectures

Abstract A403

keywords: agent architectures, reactive control, hierarchical planning, situated reasoning

Abstract

Within the agent architecture community, there is a controversy as to whether behavior organization must be fully dynamic, or can be more rigorously structured. We use Tyrrell's (1993) Simulated Environment to compare his fully parallel ERP action-selection mechanism with another architecture, Edmund, which supports sequences and selective attention within control. We demonstrate that the latter strategy can be more effective, and argue that this implies managing complexity is more important than preserving full reactivity for intelligent agent control.

Introduction

The use of both hierarchical and fixed sequential orderings of behavior for action selection has been postulated since the time of the early ethologists (Lorenz 1973). The advantages of such systems are clear: they reduce the combinatorial complexity of control. The set of actions which may be selected from is determined by context, including state internal to the agent. Once an agent is attending to a particular strategy, actions associated with alternative strategies are unlikely to interfere with behavior production.

The first major approach to action selection in artificial intelligence has been generative planning. Given an an initial state, a goal state, and a set of operators, plans were constructed through various forms of heuristic search (Georgeff & Lansky 1987; Newell & Simon 1981). These plans were primarily sequences of steps, though they could also be hierarchical, defined in terms of subtasks. Because of the computational expense due to the combinatorial complexity of these search-based methods, such plans are difficult to construct in real time. Consequently, they perform poorly in dynamic environments.

Another approach, reactive planning, reverses the emphasis of traditional planning, focusing primarily on the agent reacting appropriately to features of the environment. As a result, the ability to generate novel plans is compromised. As initially conceived (Brooks 1991), the only variation in the behavior of a reactive system is a consequence of its external sensing. The designer of a reactive system is responsible for providing appropriate behaviors for any significant environmental contingency. Reactive plans also exploit hierarchy: low-level sensing and action routines are combined into units referred to as *behaviors* which reduces complexity for the designer.

A basic premise of much work in reactive control is that, in order for a system to be truly reactive, all of the behaviors in the system must be constantly active (Hendriks-Jansen 1996; Tyrrell 1993; Brooks 1991; Maes 1991). The behaviors are the perceptual system of the agent. If they are inactive, relevant information in the environment will be ignored. Some authors also reject hierarchy in a controller altogether (Hendriks-Jansen 1996; Maes 1991), on the basis that it results in staged, responses governed and unreactive behavior. Others design systems that exploit hierarchical order, but still maintain constant parallel processing.

This paper compares one such architecture, Tyrrell's ERP, with another hierarchical architecture which is not fully parallel. We developed the latter architecture, Edmund, while exploring the issues of scaling behavioral complexity in reactive systems. It has two structural elements for organizing behavior: a flexible hierarchy only the roots of which operate in parallel, and inflexible action sequences. Edmund was developed for controlling autonomous mobile robots, and has been implemented on a Nomad 200 (author 1997). We compare these two architectures in the simulated environment described in (Tyrrell 1993). Five architectures, including the erp and (Maes 1991) had previously been tested in the same environment. Consequently, we are able to compare the behavior generated by Edmund with both hierarchical and non-hierarchical parallel systems.

Background

Although Edmund was developed for robot control and has already been used in robot experiments, this paper describes experiments implemented in Toby Tyrrell's Simulated Environment (SE). There were several advantages to this approach:

- A significant target architecture for comparison, Tyrrell's Extended Rosenblatt and Payton Architecture (ERP), has already been implemented by its author in this environment. This eliminates questions

of incorrect implementation of the architecture to which we are comparing our own, and saves development time. Even where full code is available for robot architectures, exact replication of results on a physically different robot is difficult.

- Tyrrell's SE provides for a larger number of often conflicting goals then most tasks easily evaluated on a robot. This is for two reasons. First, simulation allows for more sophisticated perceptual categorization than is normally achieved with robot sensing, which in turn allows for more complex task descriptions. Second, the units of behavior selected are relatively high level, allowing for greater diversity of behavior in relatively fewer actions.
- Simulation facilitates more extensive testing, which allows statistically significant results in comparing architectures even when the task has a very large number of random variables, giving a wide variance of test results even within a single architecture.

The Simulated Environment

(Tyrrell 1993), specifies an action-selection task based in a simulated environment which requires an architecture to manage a large number of conflicting goals. He defines an environment in which a small omnivorous animal is required to survive and breed. He defines six subproblems for the animal to solve.

1. Finding sustenance. In addition to water, there are three forms of nutrition, satisfied in varying degrees by three different types of food.
2. Escaping predators. There are both feline and avian predators, which move at different speeds and have different perceptual capabilities.
3. Avoiding hazards. Benign dangers in the environment include wandering herds of ungulates, cliffs, poisonous food and water, temperature extremes and darkness. The environment also provides various forms of shelter including trees, grass, and a den.
4. Grooming. Grooming is necessary for homeostatic temperature control and general health.
5. Sleeping at home. The animal is blind at night; its den provides shelter from predators and other hazards, and also helps the animal maintain temperature while conserving energy.
6. Reproduction. The animal is male, thus its reproductive task is reduced to finding, courting and inseminating mates. It is hazardous to try to inseminate unreceptive mates. The success of the animal is determined by the number of times it mates in a lifetime.

These problems vary along several axes: homeostatic vs. non-homeostatic, dependency on external vs. internal stimuli, periodicity, continual vs. occasional expression, degree of urgency and finally, whether prescription or proscription with regard to particular actions.

In addition to these problems, the environment is very dynamic —food quantities, temperature and light vary, animals move — and sensing and action uncertain. Perception in particular is extremely limited and severely corrupted with noise; the animal usually misperceives anything not immediately next to it, unless it has chosen to expose itself by looking around in an uncovered area.

Tyrrell separates the problems of learning and navigation from the problem of action selection by providing these elements in his simulation. Thus the animal has available as primitives a direction in which it thinks it remembers its home or recently observed food and water. The animal's sense of location with respect to its den decays over time and distance, thus keeping track of its bearing is a part of the "sleeping at home" sub-problem, even though the precise mechanism of navigation is not explicit.

The results of the animal's performance is heavily dependent on chance implementation details of the environment. Tyrrell attempts to check the independence of his from these parameters by running all his models in four different "worlds". Besides the standard model he first designed, there are three variants. As documented (p. 162) these vary in the following ways.

1. Perception is altered by affecting the animal's visibility according to the time of day, the amount of vegetation, and the animal's own activity.
2. The noise variance for navigational information is tripled, and the size of the remembered map is halved.
3. Motor control is compromised by making it more probable that incorrect actions are be taken, and by changing the conspicuousness and energy consumption of various actions.

The Extended Rosenblatt and Payton Action Selection Mechanism

(Tyrrell 1993) implements and tests four theories of action selection, often extending and providing necessary and obvious modifications to the theories in the attempt to make them operate on a real system. The first was drive theory ("simple motivational competition") as described by Hull, the second combined Lorenz's two "Psycho-Hydraulic" models, the third implemented Maes' spreading activation networks, and the fourth Rosenblatt and Payton's "connectionist, hierarchical, feed-forward network." Tyrrell found an extension of this last theory to perform best in his SE.

The Rosenblatt & Payton model had, like Edmund, been developed and proven as a robot architecture (Rosenblatt & Payton 1989). The main feature of their architecture is what Tyrrell terms a *free-flow hierarchy*. Nodes in the hierarchy receive activation from internal

and external stimuli and higher elements of the hierarchy. They pass energy to their children. The ultimate leaf nodes of the tree are actions, which are activated on a winner-take-all basis.

What differentiates the ERP from the drive model or other hierarchies is that no decisions are made until the leaf or action nodes. This allows for the selection of *compromise candidate* behaviors: behaviors that satisfy or express multiple drives. See for example Figure . Tyrrell made three primary extensions to Rosenblatt & Payton's original architecture:

- He added penalties for temporal delays and uncertainty of rewards for items that are distant or based on poor memory or perception. This allows the animal to chose a small but apparent near food source rather than a large but distant, poorly remembered food source.
- He created explicit penalties for appetitive vs. consummatory actions. This was also a problem for Maes' architecture in the SE, both architectures would arbitrarily pass up opportunities to fulfill real needs, in other words, they would dither. Maes architecture performed particularly poorly in the SE.
- Finally, Tyrrell created a new rule for combining the inputs to a single node that explicitly distinguished the maximum of inputs adding positive energy to a particular node and the minimum of inputs opposing the node. This was significantly more successful than simple summation of the activations.

Tyrrell's architecture work has been very influential. See for example (Blumberg 1996) for a recent architecture concerned with combining influences of various behaviors, and (Humphrys 1996) for an exploration of action-selection mechanisms that consider positive and negative opinions in negotiating action selection between various goals or drives. We chose to evaluate the ERP directly not only because of the provision of the SE and the established comparisons with Maes and the Drive theories, but also because the ERP is a good example of the kind of more dynamic architecture theoreticians such as (Hendriks-Jansen 1996) expect to do better than relatively discrete architectures such as Edmund.

Edmund

Edmund is intended to extend behavior-based approaches to artificial intelligence, such as subsumption architecture (Brooks 1991) to allow for behavior of greater complexity. Subsumption architecture stipulates that intelligence should be decomposed into individual behaviors, each of which controls its own sensing and activation. All behaviors run continuously in parallel. Whether a behavior is expressed in the control of the robot is determined partly by the behavior itself, which can recognize situations in which it should be active, and partly by intricate interactions with other behaviors. Behaviors are allowed to inhibit

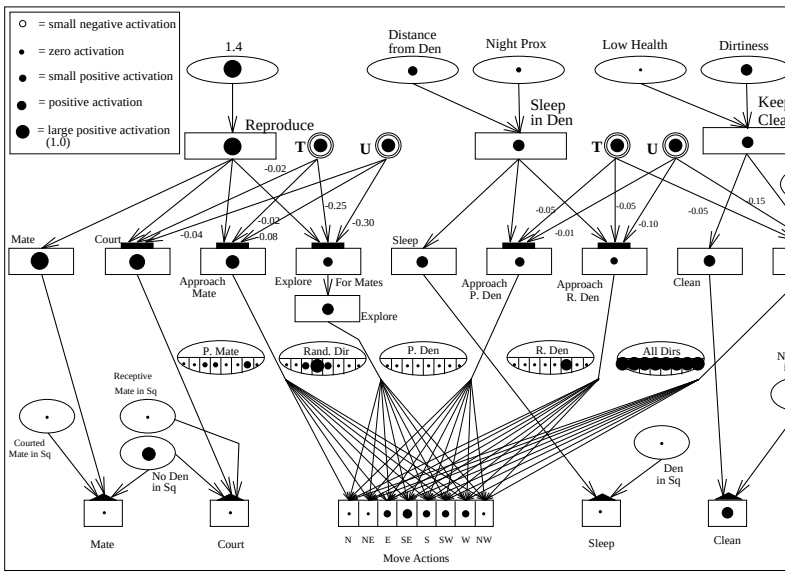


Figure 1: A fraction of the Extended Rosenblatt & Payton action selection network for controlling an animal in Tyrrell's Simulated Environment. **T** and **U** are temporal and uncertainty penalties for actions that take time or pursue a goal uncertain to succeed. P stands for perceived, R for remembered. From (Tyrrell 1993), page 247.

sorry about width, will fix!

and change each other's input, and also to suppress each other's output. Behaviors are organized in carefully crafted networks that specify these interactions. Groups of behaviors for pursuing the same goal are implemented as a unit called a level. Levels of additional, higher goals are added in a linear hierarchy. The behaviors in higher levels may inhibit or suppress lower levels' behaviors, but lower levels are intended to remain "unaware" of higher. The lower levels' behaviors should be unchanged from their internal perspective.

Subsumption architecture works well in situations where there is a small number of non-conflicting goals that require behaviors that are fairly homogeneous across the entire of the robot's intended niche (Horswill 1993; Bryson 1995). However, it is difficult to specify the performance of different behaviors in the same physical environment, such as when a sequence of behaviors needs to be conducted, or when the robot is selectively pursuing one of multiple possible goals (Connell 1990). This problem is addressed in Edmund by creating a more complex hierarchical control system for triggering the expression of behaviors. This structure is based on two elements: the *competence* and the *action pattern*. An action pattern is a sequence of behaviors that will execute in turn unless interrupted by a radical failure or a higher priority goal. A competence is a grouping of action pat-

terns and other competences which express a behavior consistent with a particular goal. The elements of a competence are prioritized so that if multiple elements are able to run, the element closest to achieving the goal is selected. Competences are similar to reactive plans (Agre & Chapman 1988), or triangle tables (Nilsson 1984; 1994). The idea of using hierarchies and sequences to improve behavior-based control is also widespread (Firby 1987; McGonigle 1991; Gat 1991; Tyrrell 1993; Correia & Steiger-Garção 1995; Levison 1996), though still controversial (Hendriks-Jansen 1996; Maes 1991).

Edmund differs from other behavior-based architectures in two ways. First, it completely separates the issue of flow-of-control from flow-of-information. The only information available in the control hierarchies is recent control and decision state. All perception and sensory-based memory is present in the behaviors themselves. The flow of information between behaviors is not restricted by the control architecture, but rather specified within a *behavior library*. Adaptation, both in terms of short term perceptual processes and long term "lifetime learning," occur in the behavior library. The variable state needed for learning is stored in the behaviors that use it.

The second difference distinguishes Edmund from other behavior-based architectures that exploit plan-like structures for organizing complexity, such as RAPs (Firby 1987) and the other hierarchy-based architectures mentioned above. Although hierarchical, the control architecture supports both reactivity and arbitrary chains and loops of behavior. This is achieved through several methods:

- Each element of a competence is prompted to perform by its own perceptual requirements. There is no error passing or assumption of success for an operation. An action sequence will complete regardless of the success of its elements, the result is checked at the competence level. Infinite loops are prevented and alternative strategies enabled by a system of habituation.
- The top level of the hierarchy consists of special competences called *drives*. Drives execute all their elements in course-grained pseudo-parallel with best-attempt scheduling for the frequency of attention to the varying elements. This allows for the constant monitoring of important sense modalities or internal states and the possibility of interruption for urgent or opportunistic conditions.
- No stack is maintained as control traverses the hierarchy of competences. When a competence prompts the activation of one of its constituent competences, the new competence takes its place in receiving the attention of the parent drive. This enables infinite loops in behavior, such as oscillating between active exploration and grooming/memorizing phases of behavior in rats exploring new enclosures (Wilson & McNaughton 1994). Drives maintain a record of there initial elements, so that if a competence chain completes, the chain may be started again from the beginning. If nothing has changed in the world, control will return to the parent of the competence that just completed which allows the parent in turn to proceed. If something relevant *has* changed, control may switch to a more appropriate competence or action pattern.

Examples of Edmund control hierarchies are included in the following section on Results, below.

Edmund thus has the potential for expressing fairly dynamic behavior, but it does not have the complete opportunism or full democracy of Maes' and Tyrrell's architectures. Modularity between control abstractions may be violated in the behavior library: there is no restriction on being influenced by the state of other behaviors. As Tyrrell and Blumberg describe, such violations are ubiquitous in animal behavior. On the other hand, Edmund allows for the expression of both sequences and prioritization. It allows for the utilization of selective attention to ensure particular tasks are completed fluently.

Experiments

Development

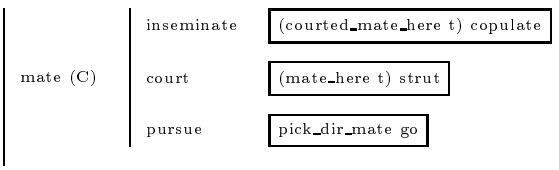
An agent architecture has been defined by Maes as "[a] particular methodology for building [agents]" (cited in (Wooldridge & Jennings 1995).) An important aspect of methodology is not only the maximal performance that might be achieved under that architecture, but also the extent to which a methodology actually facilitates the development of intelligence (McGonigle 1991; Bryson 1998). Thus in comparing architectures, ideally there would be some control made for the amount of time spent in development and the ability of the programmer or programmers involved. Of course, this is very difficult across projects. Tyrrell states that he made an attempt to spend approximately the same amount of time adjusting parameters for each of the architectures he implements, but does not mention the amount of the time spent.

The experiments documented here were carried out by a single programmer, with a professional proficiency in C and C++. Tyrrell's thesis was studied carefully and a plan for programming Edmund developed on a part-time basis over the period of a week. Compiling and testing Tyrrell's work, and connecting the new C++ library code for Edmund to the SE took approximately 5 days, full time. Developing the results described below took approximately 18 days over two separate periods (12 and 5 days each), due primarily to the time for running statistically significant trials for the tweaking of parameters and the testing of algorithms. Actual time spent coding was approximately 4-6 days. The large number of trials necessary for both development and statistically significant comparison is

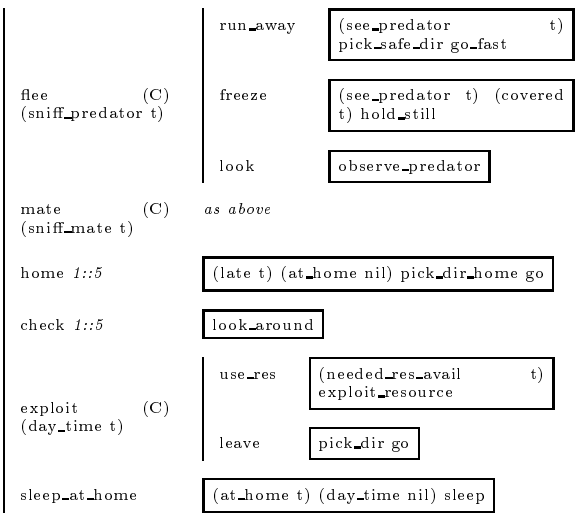
a consequence of the high variety of environments created by the SE.

Implementation

We began with the most basic program that could generate a genetic fitness measure – one that only pursued mating. Thus the drive "life" had a single competence "mate", which consisted of:



This simple program performed fairly well. Though on average it lived under half a day, it had a genetic fitness of slightly more than 2, immediately outperforming Tyrrell's best implementation of (Maes 1991). Other obvious behaviors were added gradually, testing the new additions and ordering on approximately 800 runs. Simplicity was favored unless the program improved very significantly. After this phase of development the components of life in order of priority were:



Homing in the evening and looking for predators take priority intermittently, but habituate to allow other, lower priority behaviors to express themselves immediately after they had fired.

At this point, the genetic fitness of the Edmund animal was approximately 65animal's performance step-by-step in the graphic user interface[1]. This resulted in the following observations and changes:

- The animal seemed too concerned with staying in shelter and failed to explore much unless chasing mates. This was altered by reducing the desire for shelter in pick_dir.
- The animal was often killed by feline predator's while freezing. Consequently we added a prohibition to the freezing action pattern that prevented freezing in the presence of felines.
- Once the animal was less concerned with staying in shelter, it sometimes explored to such a degree that it lost track of its bearing entirely and could not recover by nightfall even though it began looking for its den with plenty of time. Consequently we added the action pattern: triangulate (lost t) pick_dir_home go

  which had the same priority as homing above, but was active all day and did not habituate. Thus, once the animal becomes slightly lost, it goes towards its den until it recognizes where it is (unless it happens across a mate or a predator.)
- The animal often had unforced accidents with irrelevant animals and cliffs, particularly while it was pursuing mates. We altered pick_mate_dir to increase the avoidance of these problems.

Some observations were ambiguous. For example, because the animal had very erratic perception, it frequently hallucinated a predator nearby. Attending to any trace of a predator disrupted behavior coherence sufficiently to interfere with mating, while paying too little attention to predators reduced mating opportunities by reducing the expected lifespan. We addressed this problem by running a large number of simulations with randomly selected values for six critical thresholds: four for detecting predators (observing or fleeing each type), one for seeking shelter and one for avoiding dangerous places.

Linear regression on this data was uninformative, because the variables relationships to the animals success were neither linear nor independent. Instead, we selected the variable sets for the top few performing individuals out of 4400 trials, and tested each over 800 additional runs. The best set improved the animals mean performance by 25 percent.

Edmund and the ERP were tested both using the parameter settings of the SE utilized in (Tyrrell 1993) and under another set of parameters with considerably sparser food supplies. We developed this latter test on the theory that it would be a greater test of persistence, since the normal SE settings provided sufficient food that opportunistic grazing in place is sufficient to maintain energy. Under the above control system, Edmund out-performed the ERP in this sparse world, except for variant world 3. Examination of the simulation code disclosed that the largest variation in this world was a very significant reduction in energy consumption for certain activities (such as mating and running) and significant changes in the conspicuousness of some behaviors, both positive and negative. We guessed our strategy was wasting too much time or conspicuousness on grazing. Consequently we divided the **exploit** competence and its elements into

[1] In approximately 50 animal life-times, which took about 5 hours to observe while making corrections.

constituent parts, actually eliminating the one case of "compromise candidate" determination in our architecture. We also changed the flight from predators to be the same as normal cautious exploration, only faster. These changes resulted in a system that surpassed Tyrrell in both the sparse food and the original versions of the SE, excepting variant world 3 to which it seems particularly well adapted.

Results

Edmund performed better (that is, mated more), but not significantly so (Critical Ratio = 1.36) in Tyrrell's world. However, it was very significantly better than Tyrrell in every world except for variant world 3, and also than any other architecture reported in (Tyrrell 1993). The final results (over 6600 trials) were:

World	Edmund	Tyrrell	CR
Standard	9.12 (0.19)	8.09 (0.17)	3.95^{**}
Var. 1	4.02 (0.09)	3.61 (0.09)	3.1^{**}
Var. 2	9.67 (0.2)	8.16 (0.16)	5.73^{**}
Var. 3	11.23 (0.23)	13.38 (0.23)	-6.58^{**}

where the parenthetical numbers indicate standard error, or the standard deviation of the mean scores.

As mentioned in the previous section, both architectures were also tested in an additional set of worlds which were identical to Tyrrell's except that food was significantly scarcer. The range of possible initial food supplies was modified as follows:

Food	Original	Sparse
fruit	[50, 81]	[5, 35]
cereal	[45, 26]	[2, 27]
prey	[25, 36]	[2, 15]

which resulted in the following reduction in performance:

World	Edmund	Tyrrell	CR
Standard	8.17 (0.19)	4.77 (0.12)	15.01^{**}
Var. 1	3.56 (0.09)	2.46 (0.06)	9.59^{**}
Var. 2	10.79 (0.18)	4.56 (0.12)	27.6^{**}
Var. 3	10.74 (0.24)	12.53 (0.23)	-5.47^{**}

As can be seen, Edmund also coped substantially better than the ERP with this more difficult situation, though again with the exception of the third variant world.

Another interesting result is the complexity of the final systems. Edmund required 20 action primitives and 22 dedicated sensing primitives. It also had 7 competences and 23 action sequences defined in its final program script. The ERP had 61 sensing nodes and 278 other nodes, of which 216 were intermediate. Edmund had 26 thresholds embedded in its primitives, of which 6 were not one of four standard values (0.2, 0.4, 0.6 or 0.8). The ERP had 264 weights, of which 189 were either 1.0 or -1.0. However, the other 75 weights took 37 separate values.

Discussion

We have been able to out-perform a carefully designed parallel architecture architecture using a system that patently ignores most of its sensory input. Arguably, this is partly due to chance. Our system was very sensitive to some parameter tweaking, particularly the threshold at which predators were attended to. Also, we had the advantage of following Tyrrell's thesis and having a target value to surpass. Nevertheless, both the relative lack of complexity of our architecture and its relative success we take as evidence that managing complexity through selective attention is more important than having a fully reactive architecture.

This result is in some sense obvious, particularly when considering other related sciences. Research in statistics, information theory and signal processing, for example, tell us that reducing information, or separating relevant information from noise, is key to intelligent processing. Animals not only control for what information is worth attending to, but when. Ethology has long had the example of digger wasps that treat caterpillars as food only when they are engaged in appetitive behavior (Lorenz 1973). Rats also will not engage in any appetitive behavior (such as foraging or eating) without the appropriate triggering of their limbic system (Carlson 1994).

The Role of Design in AI

We doubt that simply the loss of information is itself sufficient or necessary to the improved behavior. Rather, it is more likely that ignoring some information reduces the complexity of the design problem, and thus improves the probability of developing a better system. The design problem does not exist simply because we chose to use hand-coded architectures for attacking the SE. The biological results mentioned above indicate that evolution may also find this "design problem" most easily solved by enforcing partial modularity.

Although the problem of "tweaking" parameters in some sense begs for automated learning of action selection, thus far no adaptive system besides natural evolution has mastered tasks near the complexity of Tyrrell's SE. Natural evolution is a very slow process that is also hierarchical — it leverages previous solutions and existing processing capabilities. Some element of design is present in any AI system, in terms of the choice of primitives, the size of the variable space, and the exact mechanism for learning. Besides the behavior-based approach, many users of planning systems such as PRS (Georgeff & Lansky 1987) capable of generative planning actually encode most of their behavior in pre-designed plans.

We see hand crafted plans as providing the bias necessary for any current system to perfect intelligent control. In related work, we are exploring the roles of specialized learning systems (such as perception and episodic memories) in agent control, and in the fu-

ture we are also interested in the relationship between generalizing hand-coded reactive controllers and case-based reasoning.

The SE Simulation as a Benchmark

One premise of this research is that the SE is a valid benchmark for comparison of action-selection mechanisms. Our general stance is that no individual benchmark could be sufficient for a thorough evaluation of a system, nor is one likely to embody all that is "hard" about intelligence. In particular, there is considerable danger of general claims about AI being made from systems designed explicitly for a particular simulation (Hanks, Pollack, & Cohen 1993). Both the Rosenblatt and Payton architecture and Edmund had been developed on robots. No serious revision was made to Edmund in the course of these experiments other than the discovery of some slow memory leaks and errors in the script parser.

The strengths of Tyrrell's environment are outlined above under Experimental Design. It certainly involved satisfying more conflicting goals and a greater number of competences than any previous task Edmund had been applied to. On the other hand, it seems particularly biased for reactive behavior, at least in its current parameter configuration. We never found it necessary to attend to any remembered food, nor to implement any cycles in the competence tree nor any action sequences longer than three elements. We have used cycles both to encode simulated infant reaching behavior and to imitate rat exploratory behavior. Creating a controller for visual routine theory experiments involved creating action patterns as long as eight elements. In other words, the SE is a reasonable mechanism for testing persistence and goal juggling, but finer-grained behavior such as real perceptual control or learning probably require other forms of complexity.

Conclusion

This paper has presented work indicating that even (or perhaps especially) in an extremely dynamic and dangerous environment, an agent may be better off ignoring some of the information available to it. We suggest this may be because finding the correct design, whether by hand-coding or through learning, is the key bottleneck to developing intelligence, and reducing information reduces the complexity of the task. We therefor argue against full parallelism and reactivity, although our winning solution displayed moderated elements of each.

References

Agre, P. E., and Chapman, D. 1988. What are plans for? AI memo 1050, MIT, Cambridge, MA.

Blumberg, B. M. 1996. *Old Tricks, New Dogs: Ethology and Interactive Creatures.* Ph.D. Dissertation, MIT. Media Laboratory, Learning and Common Sense Section.

Brooks, R. A. 1991. Intelligence without representation. *Artificial Intelligence* 47:139–159.

Bryson, J. 1995. The reactive accompanist: Adaptation and behavior decomposition in a music system. In Steels, L., ed., *The Biology and Technology of Intelligent Autonomous Agents.* Springer-Verlag.

Bryson, J. 1998. Agent architecture as object oriented design. In Singh, M. P., ed., *The Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL97).* Springer-Verlag.

Carlson, N. R. 1994. *Physiology of Behavior.* Boston: Allyn and Bacon, 5 edition.

Connell, J. 1990. *Minimalist Mobile Robotics: A Colony-style Architecture for a Mobile Robot.* Cambridge, Massachusetts: Academic Press. also MIT TR-1151.

Correia, L., and Steiger-Garção, A. 1995. A useful autonomous vehicle with a hierarchical behavior control. In Moran, F.; Moreno, A.; Merelo, J.; and Chacon, P., eds., *Advances in Artificial Life (Third European Conference on Artificial Life)*, 625–639. Berlin, Germany: Springer-Verlag.

Firby, J. 1987. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 202–207.

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

Georgeff, M. P., and Lansky, A. L. 1987. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, 677–682.

Hanks, S.; Pollack, M. E.; and Cohen, P. R. 1993. Benchmarks, testbeds, controlled experimentation and the design of agent architectures. Technical Report 93–06–05, Department of Computer Science and Engineering, University of Washington.

Hendriks-Jansen, H. 1996. *Catching Ourselves in the Act: Situated Activity, Interactive Emergence, Evolution, and Human Thought.* Cambridge, MA: MIT Press.

Horswill, I. D. 1993. *Specialization of Perceptual Processes.* Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Humphrys, M. 1996. Action selection methods using reinforcement learning. In Maes, P.; Matarić, M. J.; Meyer, J.-A.; Pollack, J.; and Wilson, S. W., eds., *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior.* Cambridge, MA: MIT Press.

Levison, L. 1996. *Connecting Planning and Action via Object-Specific Reasoning.* Ph.D. Dissertation, University of Pennsylvania. School of Engineering and

Applied Science, Computer and Information Science Department.

Lorenz, K. 1973. *Foundations of Ethology.* New York: Springer-Verlag.

Maes, P. 1991. A bottom-up mechanism for behavior selection in an artificial creature. In Meyer, J.-A., and Wilson, S., eds., *From Animals to Animats*, 478–485. Cambridge, MA: MIT Press.

McGonigle, B. 1991. Incrementing intelligent systems by design. In Meyer, J.-A., and Wilson, S., eds., *From Animals to Animats*, 478–485. Cambridge, MA: MIT Press.

Newell, A., and Simon, H. A. 1981. Computer science as empirical inquiry: Symbols and search. In Haugeland, J., ed., *Mind Design.* Cambridge, MA: MIT Press. chapter 1, 35–66.

Nilsson, N. 1984. Shakey the robot. Technical note 323, SRI International, Menlo Park, California.

Nilsson, N. 1994. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research* 1:139–158.

Rosenblatt, K., and Payton, D. 1989. A fine-grained alternative to the subsumption architecture for mobile robot control. In *Proceedings of the IEEE/INNS International Joint Conference on Nueral Netowrks.*

Tyrrell, T. 1993. *Computational Mechanisms for Action Selection.* Ph.D. Dissertation, University of Edinburgh. Centre for Cognitive Science.

Wilson, M., and McNaughton, B. 1994. Reactivation of hippocampal ensemble memories during sleep. *Science* 261:1227–1232.

Wooldridge, M., and Jennings, N. R. 1995. Intelligent agents: Theory and practice. *Knowledge Engineering Review* 10(2).