Joanna Bryson's Ph.D. Proposal

Signed draft, of May 3, 2000. Please do not distribute without permission.

1 Thesis

Work in agent-based artificial intelligence has lead to an understanding of the problems and solutions involved in designing complex, animal-like intelligence. This dissertation presents a significant formalization of these techniques with contributions in both methodology and understanding. The core contributions of the dissertation is a methodological approach, Behavior Oriented Design. This strategy extends the behavior-based approach to artificial intelligence, incorporating methodology from Object Oriented Design. Its advantages over standard behavior-based techniques are an improved methodology in determining behavior decomposition, managing and exploiting learning, and addressing the problem of behavior arbitration. The BOD methodology may be thought of as an agent architecture; it may also be implemented as an extension or refinement on top of most major software architectures. Examples of both approaches are described in this dissertation.

Evidence supporting the BOD methodology is drawn from successes and failures demonstrated in the agent literature of the last 15 years, and through several novel implementations. The principle demonstrations in this dissertation are in the domain of natural language and social learning. This domain was chosen both because it allows for a large range in scaling of cognitive complexity, and because natural language in particular has an enormous research community — consequently many of the basic behavioral elements required are available in the public domain. Depending on the progress of the project and available resources, there may also be demonstrations of non-linguistic, ape-like social learning in VR, or possibly robotics. Previous work in robotics and VR will also be discussed.

2 Proposed Dissertation Structure

The dissertation can be decomposed into three parts. The first part introduces the main hypotheses and supports them with both reason and evidence from literature. The second part presents the recommended methodologies in detail. The third part demonstrates the methodologies with several examples of research using systems built in this methodology. Here are the proposed chapter titles:

Working Title: **Intelligence by Design**

1. **Introduction**
2. **The Place of Learning**
3. **Fitness in Agent Architectures**
4. **Behavior Oriented Design**
5. **POSH Action Selection**

6. **Design of a Dialog System**
7. **Specialized Learning in a Dialog System**
8. **Non-Linguistic Social Learning**
9. **Conclusions**

3 Outline of the Remainder of the Proposal

The following sections describe the main arguments of this thesis. They are followed by several schedules showing how the research and dissertation are intended to be completed over the next 10 months. These include a calendar, a hacking schedule, a list of papers intended to make up the thesis, and references to those papers which already exist.

4 Introduction

The introduction will discuss the relevance of design issues to AI, and how they relate to learning. It will then introduce the design methodologies and examples of their utility, though these are currently scattered in the descriptions of other chapters below. The introduction concludes with a discussion of the approach by which the dissertation's theses are argued.

4.1 Design

Hand design or programming of systems has probably always been the dominant means of creating AI systems. The intrinsic difficulty of hand coding has lead to a good deal of research into alternate strategies such as learning and automated design. Machine learning in particular has been successful for particular tasks, such as recognizing credit card fraud or designing control systems for chemical plants. However, these successes mostly address problems of pattern matching, and do not solve the problem of complex intelligent control. By complex intelligent control, I mean control capable of producing ordered behavior in complex situations that allow for multiple actions and the pursuit of multiple goals.

Planning is the area of AI that has grown to solve these sorts of problems. Originally, planning was intended to determine the sequence of steps necessary by performing logical analysis of the problem and search for a correct sequence. Although this approach has seen considerable advances in the complexity of the tasks it can tackle (see Weld, in press), this complexity is still far below the level required for applications that operate in real time in dynamic worlds containing multiple, independent actors and multiple, conflicting possible goals.

This problem of complexity has been proven intractable in the limit case (Chapman, 1987). Further, there is strong evidence that in the average case it is also intractable. This evidence can be found in the research trends of the planning field. AI users working under a many different paradigms have turned repeatedly to designing their plans by hand. These new approaches are usually collectively identified as *reactive planning*. A reactive planner is a system that selects only its next move at any particular time interval, and does this selection based on an application of information about its current context to a store of provided code — a reactive plan. Although there has been some research into automatically generating reactive plans (e.g. Schoppers, 1995; Benson, 1996), most applications rely on human designers creating the reactive plans (e.g. Georgeff & Lansky, 1987; Reilly, 1996)[1].In fact, the original meaning of "planning" described above has now been qualified, it is now referred to as *constructive planning*

The success of these approaches indicates that the problem of intelligence is simply too combinatorially complex to be solved by search algorithms such as constructive planning and reinforcement learning unless there is an enormous quantity of time (as in the case of biological evolution) or there is an enormous quantity of scaffolding or bias (as in the case of biological development.) This argument is not intended to say that learning or development by the functioning agent may not be integral parts of developing an intelligence. Learning can greatly facilitate design, just as it can facilitate evolution (Turney et al., 1996; Hinton & Nowlan, 1987). What it does say is that design is necessary to in turn enable these processes, at least in reasonable time frames for developing artifacts.

4.2 Methodologies and Mechanisms for Agent Design

Agents are generally designed to the specification of an agent architecture. An agent architecture can consequently be thought of as a design methodology: its purpose is to assist developers in providing the bias or scaffolding necessary to enable the development of a complex intelligence (Wooldridge & Jennings, 1995; Maes, 1991). Agent architectures generally consist of some specific mechanisms which contribute to this function. For example, many architectures provide scripting languages for reactive plans. The agent designer is expected only to write a simple script, which a program provided by the architecture will then execute. However, a mechanism such as this should not be confused with the complete architecture. Most popular architectures have several implementations of their core mechanisms, while others have no single mechanism and are rather used as gross guidelines for agent programmers.

Further, the provision of a mechanism should not be confused with the provision of a full architecture. There are many levels and details in designing an agent, and while no architecture can yet provide a full specification, architects should not be too jaded to the requests of their users for more fully specified methodologies.

[1] An interesting corollary I need to track down a reference on — apparently most of Soar's users never use the learning features that are the theoretical basis of the architecture, but hand code all the productions and turn the learning off, so it doesn't disturb their programs.

4.3 Structure of a Methodology Thesis

Although I have argued above that work on methodology is extremely important to artificial intelligence, it is also not an easy topic for an engineering dissertation. Assertions about ease of use usually cannot be proven in the mathematical sense of the word. Further, statistically valid scientific evidence demonstrating significant improvement is difficult and expensive to come by: it requires a large sample of programmers tutored in a variety of methodologies, and the comparison in terms of time and effort and quality of final product of the outcome of their programming. Consequently, most architecture theses resort to combining plausible arguments with a demonstration of some impressive system or systems constructed under the methodology.

In the case of this dissertation, although I have employed both of these strategies, I have also attempted to add a more scientific approach. Rather than hiring a large number of programmers myself, I examine the history of agent design as made available in the literature. When practitioners from various different paradigms of AI research have converged on a particular methodology, I take this as evidence of the viability of that method. This is particularly true when the paradigm began in stated opposition to a particular methodological aspect, or when a methodology has been subjected to substantial application with significant results. In these cases the selection of the methodological aspect can reasonably be attributed to forces other than simply those of personal belief or some other social bias.

This strategy of taking evidence cannot lead to perfect certainty, but it can lead to high probability. In this sense, it is very like the approach used in a scientific dissertation, particularly one employing arguments from evolution or history. These dissertations and their theses will always be more controversial than theses conclusively proven, but their approach is better than leaving important areas of research unexamined.

5 The Place of Learning

An agent's intelligence comes from two sources: knowledge derived from its own experience, and knowledge provided as part of its embodiment. For an animal, this provided knowledge has been derived by evolution; for an artifact, it must be provided by the developer. The relationship between these two knowledge sources is complex — the capacity for individual learning must itself be provided, but the evolution or development of provided knowledge can be sped substantially if it is aided by individual learning (e.g. the Baldwin effect).

From the perspective of designing behavior, provided knowledge and individual learning are theoretically interchangeable. Small amounts of adaptive state can radically reduce the number of states needed to represent provided behavior plans. Examples of such minimalist state include hierarchically structured reactive plans which maintain decision state or a small number of perceptual "pointers" as in deictic vision algorithms. Some knowledge is difficult to anticipate prior to the assembly of an agent, for example exact perceptual experience or actuator control. In these situations, tightly constrained unsupervised learning techniques may be used to reliably achieve the classification and discrimination necessary for successful behavior.

This chapter supports these arguments with examples from both artificial and natural intelligence literature, and illustrates the case with examples from my own research, including the blocks world work done at MIT in 1995 and the research presented in my MPhil.

6 Fitness in Agent Architectures: Evidence from the Literature

In this dissertation I argue that there are three important components for an agent architecture for complex agents in dynamic worlds. They are:

- Modularity (though not necessarily full encapsulation). Decomposition is a well established strategy for simplifying the design problem by reducing the complexity of the individual elements to be designed. It also facilitates parallelism, discussed below.
- Structured action selection. Modularity leads to difficulties of behavior arbitration, or action ordering. Privileged representation of sequential and hierarchical behavior structure ("reactive plans") is a mechanism for focusing attention and combating combinatorics.
- Parallelism. Specialized parallel mechanism which can recognize high-priority situations and switch attention to them allow an agent to survive in and appropriately exploit an unpredictable dynamic environment. Note that this strategy also requires parallel, autonomous perception.

The argument for the "three necessary components" can be supported from the AI literature, especially when the literature is supplemented by fuller project reports from the developers. The literature shows evolutionary trends indicating selective pressure towards the necessary components. This review may be extended to include production rule systems, and mammalian brains.

In my review of the agent architecture literature, I examine the development of four different paradigms — behavior-based architectures, layered / hybrid architectures, PRS (BDI), and Soar (with some ACT-R). I look at what architectures have been successful and at what architectural features have been added in response to the demands of practical systems.

I conclude that there are several necessary architectural attributes for producing an agent that is both reactive and capable of complex tasks. One is an explicit means for ordering action selection, in particular a mechanism exploiting hierarchical and sequential structuring. Such a system allows an agent with a large skill set to focus attention and select appropriate actions quickly. This sort of mechanism needs to be complimented by a parallel environment monitoring system in order for agents to be reactive. Finally, modularity seems to benefit all architectures. This is presumably because modularity substantially simplifies the design process by substantially simplifying the individual components to be built.

Modularity here refers to the decomposition of an agent's intelligence, or some part of its intelligence, into a number of smaller, relatively autonomous units. This does not necessarily imply fully encapsulated modules as in Fodor (1983), where the state and functionality of one module are strictly unavailable to others. There are actually two axes of modular decomposition, both of which are used in this dissertation. First, there is modularity decomposed along the lines of ability, with the module formed of the perception and action routines necessary for that ability, along with their required or associated state. Second, there is modularity by functionality, so for example an architecture may have special-purpose modules for action-selection or motor/position-smoothing. In biological terms this first set of behaviors is more analogous to modularity within a single organ such as the neocortex or spinal chord, while the second is more analogous to specialized organs such as the hippocampus or cerebellum.

Fully modular architectures can create new design difficulties. If sequential and hierarchical control are avoided, then specifying priority and action selection between the interacting modules becomes difficult. However, an architecture that does allow for a centralized control system to focus attention and select appropriate actions may fail to notice dangers or opportunities that present themselves unexpectedly. Agents existing in dynamic environments must have architectural support for monitoring the environment for significant changes, in order for the complete agent to remain responsive. This may either be a part of the main action-selection system, or a separate system with priority over primary action selection.

An example of the sort of evidence in this chapter is the progressive increase in control structure found in Behavior-based AI. I start with Subsumption Architecture (Brooks, 1986). Subsumption architecture began with only strictly ordered layers. Over time and experimentation some tasks were found to require the ability to focus attention on a single behavior while allowing it to execute a behavior pattern (Connell, 1990). This leads towards the ANA architecture (Maes, 1991) which has more flexible goal-ordering and plan-sequencing information available through arcs between behaviors. Next I discuss Tyrrell (1993) and his free-flow hierarchies, and then Blumberg, where the somewhat free-flow hierarchies have the ability to lock out other behaviors at the level of shared resources (Blumberg, 1996). More recent work on this architecture is more explicit about the use of hierarchy for simplifying control (Kline & Blumberg, 1999). On some long term research projects, this kind of evidence has been well documented e.g. (Nilsson, 1984; Laird & Rosenbloom, 1994). I try to point out the value of these information sources while performing meta-analysis across them.

7 Reactive Planning with POSH Action Selection

This chapter does an in depth explanation of reactive planning, and presents a particular set of idioms that are a part of the BOD methodology. I'm not actually certain whether this chapter should come before or after the BOD chapter. In particular it describes the Parallel-rooted, Ordered Slip-stack Hierarchical (POSH) action selection used in BOD.

Reactive AI initially was taken to mean "stateless AI," but this definition proved intractable. All agents had at least some program state. Reactive came to mean simply responsive and opportunistic, or simply to lack deliberation. As I argued in the first chapter, this essentially meant that some code, a reactive plan, is provided in advance. Execution of this plan is a continuous process of sensing the world and choosing the next action based on that information. Thus a single reactive plan can construct an infinite number of conventional plans.

This process of creating sequences of behavior from a reactive plan is demonstrated clearly. The reactive plan is a single element of a fundamental idiom, so fundamental in fact that I call it the "basic reactive plan," within POSH it is referred to as a competence. This is a prioritized list of elements, each with preconditions, which if executed will achieve a particular goal.

After illustrating this functionality, I discuss the utility of other attributes, such as hierarchy and fixed sequencing. I then describe attributes which ensure the reactivity of a system based on hierarchy. The first is maintaining multiple, parallel roots which are visited at the beginning of each cycle. This allows reactivity as described in Chapter 1. The second is the slip stack: limiting the stack to only the most recent one or two levels of control. If a competence completes without chaining control to one of its key elements, then action selection returns to its root, to re-evaluate the current situation. The limiting of stack keeps the system quick (reducing the overhead of hierarchy) in the case of long chains or even cycles, thus the control hierarchy does not have to be acyclic.

These attributes have been used in a stand-alone architecture, Edmund (Bryson & McGonigle, 1998; Bryson, 1999*b*), which has been implemented in several programming languages and used in several different problem domains. They have also been incorporated into other architectures: Ymir, a multi-modal character architecture (Thórisson, 1999; Bryson, 1999*a*; Thórisson & Bryson, in preperation) and PRS, a leading reactive architecture for agents and robots (Georgeff & Lansky, 1987). This work is described in more fully by (Bryson, in preperation).

Several implementations of this architecture are described, including at least two in programming languages, and two implemented within other architectures.

Finally I may address and demonstrate, explanation and foresight using reactive plans. Explanation requires "deliberate" access to the current world state, episodic memory of behavior, and the plans themselves. There will be some ambiguities, but this system should be able perform human-like explanations. Foresight would be akin to Lynn Stein's imagination work (Stein, 1994). My work on this area hasn't been done yet, it was only recently designed as an ancillary problem of the dialog tutoring system described below. I may do this in collaboration with Dave Glasspool in the context of the Norman and Shallice model of sequential actions Norman & Shallice (1986); Bryson (2000).

8 Behavior Oriented Design

8.1 Behaviors and Behavior-Based Design

Behavior Oriented Design is a methodology for constructing complex agents. It is designed to be applicable under any number of languages and most popular agent architectures. As can be gathered from its name, BOD is a derivative of Behavior-Based Artificial Intelligence (BBAI) (Brooks, 1991; Maes, 1991; Matarić, 1997), informed by Object Oriented Design (OOD) (e.g. Coad et al., 1997). Behavior-based AI is an approach that specifies that intelligence should be decomposed along the lines of perception and action. Behaviors are described in terms of sets of actions and the sensory capabilities necessary to inform them. This sensing must inform both *when* the actions should be expressed, and *how*. In other words, there are really two forms of sensing: sensing for detecting context, and sensing for parameters and feedback of motor actions.

The central observation of behavior oriented design is that mere sensing is seldom sufficient for either detecting context or controlling action. Rather, both of these abilities require full perception, which in turn requires memory. Perception exploits experience and expectation to perform discriminations more reliably than would otherwise be possible. This observation has two consequences in the BOD methodology. First, memory becomes an essential part of a behavior. In fact, memory requirements serve as the primary cue for *behavior decomposition*, — the process of determining how to divide intelligence into a set of modules. This strategy is analogous to the central tenet of object-oriented design, that process is best described and ordered in terms of state.

Although behaviors should be autonomous in so far as they provide for their own awareness of appropriateness to context, they should not necessarily be so sufficiently informed as to know whether their current operation is in line with the intentions or behavioral context of the entire agent. This is the other consequence of acknowledging the role of expectation in perception. The intentions of the agent are themselves state, and as such are the domain of a separate module dedicated to arbitration between behaviors. This process is known as action selection.

8.2 Action Selection in BOD

Action selection, unless sufficiently informed, turns into a combinatorially explosive search process, such as productive planning (Chapman, 1987). Behavior oriented design addresses this problem in two ways. First, in common with standard BBAI, the behaviors themselves control many of the details of action, thus significantly reducing the potential search space. Second, BOD relies on reactive planning. Reactive planning provides the expertise of experience and expectation in the form of preprogrammed plan elements, which are executed as seems appropriate based on the agent's perceptions. These perceptions are again based in the behaviors. As this indicates, BOD does not take the strictly encapsulated view of modularity, but rather allows for a well-defined interface between behavioral modules. The form of this interface is also taken from OOD. The interface is built of methods on the objects that represent the agent's behaviors.

As mentioned above in Section 7, reactive planning provides for the sequencing of behavior through the appropriate execution of reactive plans. Reactive plans are not themselves limited to sequential structure, but generally also exploit hierarchy, with a number of different possible plan elements ready for execution at any particular time. BOD's POSH action selection provides for both of these means of ordering behavior. *Action patterns* are simple sequences of action and sensing primitives. *Competences* are prioritized collections of plan elements, the operation of which will tend to achieve a particular goal. Most plan elements will also carry perceptually-dependent preconditions for determining whether the element can and should operate. When a competence is active, its highest priority element that can operate is activated.

In addition to these structures, some form of parallel mechanism for switching attention is necessary in order for an agent to be sufficiently reactive to operate in a dynamic world (c.f. Georgeff & Lansky, 1987; Matarić, 1997; Sloman, 2000; Bryson, in press). POSH action selection has this mechanism built into the special parallel root of the hierarchy, a special form of competence called a *drive collection*. When a higher priority drive is triggered, whether by perception or proprioception, POSH will switch full operating attention to that drive.

8.3 The Design Process

The analogy between BOD and OOD is not limited to the obvious implied metaphor of the behavior and the object, as discussed in Section 8.1, nor to the use of methods on the behavior objects for specifying the interface to the reactive plans, as explained in Section 8.2. The most critical aspect of BOD is its emphasis on the design process itself. As in OOD, BOD emphasizes cyclic design with rapid prototyping. The process of developing an agent alternates between developing libraries of behaviors, and developing reactive plans to control the expression of those behaviors. As in OOD, BOD provides guidelines not only for making the initial behavior decomposition, but also for recognizing when a decomposition has turned out to be inadequate, and heuristic rules for correcting these problems.

The Initial Decomposition The initial decomposition is a set of steps. Executing them correctly is not critical, since the main development strategy includes correcting assumptions from this stage of the process. Nevertheless, good work at this stage greatly facilitates the rest of the process.

The steps of initial decomposition are the following:

1. Specify at a high level what the agent is intended to do.
2. Describe likely activities in terms of sequences of actions. These sequences are the the basis of the initial reactive plans.

3. Identify an initial list of sensory and action primitives from the previous list of actions.
4. Identify the state necessary to enable the described primitives and drives. Cluster related state elements and their primitives into specifications for behaviors. This is the basis of the behavior library.
5. Identify and prioritize goals or drives that the agent may need to attend to. This describes the initial roots for the POSH action selection hierarchy.
6. Select a first behavior to implement.

The lists compiled during this process should be made either in a notebook, or better in computer files, which can serve as documentation of the agent. If the documentation is kept in files, the use of a revision control system is strongly recommended, in order to record the project's history.

Experience has shown that documentation is most likely to be maintained if it is in fact a functional part of the code. For this reason, code files for the separate elements of the system should be kept segregated by function. In particular, most reactive architectures require special definitions of the coded primitives. These definitions should be kept in a single dedicated file, and the primitives sorted by the behavior module in which they are implemented. This file being functional is therefore always up-to-date, and a clear and convenient documentation of the interface.

In selecting the first behavior, it is often a good idea to choose a simple, low-level priority that can be continuously active, so that the agent doesn't "die" immediately. For example, on a mobile robot with a speech synthesizer, the bottom-most priority of the main drive hierarchy might be a "sleep" function, which keeps track of the time and snores every 30 seconds or so. This way, the developer has a clear indication that the robot's control has not crashed, but that none of its interesting behaviors can currently trigger.

The Development Process The remainder of the development process is not linear. It consists of the following elements, applied repeatedly as appropriate:

- coding behaviors,
- coding reactive plans,
- testing and debugging code, and
- revising the specifications made in the initial phase.

Usually only one behavior will be actively developed at a time. Again, using revision control is a good idea, particularly if multiple developers are working on behaviors.

Reactive plans grow in complexity over the development time of an agent. Also, multiple reactive plans might be developed for a single platform, each creating agents with different overall behavior characteristics, such as goals or personality. It is best to keep all working plans in a special library, each commented with the date of its development, a description of its behavior, and a record of any other plan or plans from which it was derived. A library of historic plans can be used as a testing suite if any radical change is made to the behavior library.

Testing should be done as frequently as possible. Developing in languages that do not require recompiling, such as perl and lisp, significantly speeds the development process, though it may slow program execution time.

Revising the Specifications The most interesting part of the BOD methodology is the set of rules for revising the specifications. As in OOD, one of the main goals of BOD is to reduce redundancy. If a particular plan or behavior can be reused, it should be. If only part of a plan or an action primitive can be used, then a change in decomposition is called for. In the case of the action primitive, the primitive should be decomposed into two or more primitives, and the original action replaced by a plan element. Ideally, the new plan element will have the same name and functionality as the original action. This allows established plans to continue operating without change.

In general, the main design principle of BOD is *when in doubt, favor simplicity.* A primitive is preferred to an action sequence, a sequence to a competence. Heuristics like the above can then indicate when the simple element must be broken into a more complex one.

If a sequence sometimes needs to contain a cycle, or often does not need some of its elements to fire, then it is really a competence, not an action pattern. A competence may be thought of, and designed, as a sequence of behaviors that might need to be executed in a worst-case scenario. The ultimate (last / goal) step is the highest priority element of the competence, the penultimate the second highest and so on. Triggers on each element determine whether that element actually needs to fire at this instant. If a competence is actually deterministic, if it nearly always actually executes a fixed path through its elements, then it should be simplified into a sequence.

Competences are really the basic level of operation for reactive plans, and a great deal of time may be spent programming them. One way a competence can flag a need for redesigning the specification is by relying on large numbers of triggers. Perception should be handled at the behavior level; it should be a skill. A large number of triggers should be converted into a single perceptual primitive. Another problem can be that too many elements are added into the competence. This makes design more difficult by increasing the probability that a design fault might result in plan elements that operate against each other, unsetting each other's preconditions. More than seven elements in a competence, or difficulty in appropriately prioritizing or setting triggers, indicates that a plan needs to be decomposed into two plans. If several of the elements can be seen as working to complete a subgoal, they may be moved into another competence which replaces them as an element of the parent plan. If two or more of the elements always follow each other in sequence, they should be removed and made into an action pattern, which is again substituted into the original competence. If the competence is actually trying to achieve its goal by two different means, then it should be broken into two sibling competences which are both inserted into the competence's parent plan, with appropriate triggers to determine which one should operate.

8.4 Problems Addressed

The main problem with Behavior-Based AI to date is that it hasn't fully acknowledged the role of careful design in its success. To the extent that this problem has been acknowledged, it has largely been to justify a retreat to mechanisms of machine learning. However, to date, no system learning system has the complexity of behavior shown possible in designed ones. In creating BOD, I am proposing that the best direction forward is to pay attention to the advances of software and systems design, and adapt them to the special problem space of complex, intelligent agents.

BOD provides the best of two worlds. It supports programmers who want to build plans to direct complex sequential tasks by providing a mechanism for exploiting reactive plans, and it supports programmers who want to simplify the problems of perception and action by exploiting behavior-based modularity for specialized perception, action and learning. BOD follows good programming and systems practice: it supports rapid prototyping and changing decompositions. It expects mistakes to be made, and allows changes to be made with minimal reprogramming and changes to the interface.

Having explicit reactive plans built as part of the architecture greatly simplifies control. When one particular set of behaviors is active (say a robot is trying to pick up a teacup) there is no need to worry about the interactions of other unrelated behaviors. The robot will not decide to sit down, or relieve itself, or go see a movie unless it is at a reasonable juncture with the tea cup. On the other hand, it may drop the cup if something truly important happens, for example if it must fend off an attack from a large dog trying to knock it over. It is much easier to express this information in a reactive plan than to build complex mutual inhibition systems for each new behavior every time a behavior is added, as is necessary in conventional BBAI. In mutual inhibition or reinforcement systems, the control problem scales exponentially; with explicit plans the problem scales linearly.

Comparing BOD to BBAI It should be noted first that BOD and its competitors are methodologies, not just algorithms. In most cases, it should be *possible* to solve problems under any approach. The difference is how easy (and consequently, how likely) it is to solve problems using a particular strategy.

What BOD offers in terms of behavior decomposition over other BBAI methods is:

- A better place to start. Instead of trying to determine what the units of behavior are, the developer determines what information the agent is going to need. This is one of the chief insights from OOD.
- A better way to fix things. Unlike other BBAI approaches, BOD does not necessarily assume that decomposition is done correctly on the first attempt.

It provides for cyclic development and neat interfaces between behaviors and control.

Example of Supporting Redesign Let's pretend that Jerry is trying to control a biped walking robot. He designs a reactive plan that is a simple finite state machine. He designs four behaviors, each of which compute actuator signal for supporting the robot from perceptual information, which is dependent on the stage of the task. One of these behaviors is called "swap", which is a behavior for bringing a leg up from behind the robot and forward into an extended position.

Now, let's pretend that Jerry has a friend Alan, who realizes that the robot keeps getting brushed by large dogs while it's walking, and needs to be able to adjust its lateral weight distribution, something Jerry wasn't worrying about. Alan decides that the first point to do this is during "swap", since the robot is particularly at risk with one foot in the air and its leg not extended for support. Under BOD, Alan can either modify the behavior "swap" to always adjust for lateral motion needs, or he can replace the call to the behavior with a call to another FSM, which exploits both the original behavior "swap" and other behaviors Alan writes.

In most BBAI, if Alan modifies a behavior, he must then check that he hasn't broken dependencies in a complex network of interconnections — in fact he is denied that approach under many methodologies. In BOD, Alan can change the behavior but make certain it does not affect other elements by limiting the expression of his changes at the interface level. Or, if he believes his change should be felt in every aspect of the robot's walking that uses the "swap" behavior, he can simply cycle through the library of plans, even in their original development order, to make sure nothing has been broken.

If he takes the second approach, he will only need to test Jerry's plan with his modification. The second approach would be completely impossible in BBAI that doesn't have explicit reactive plans. It would have to be replaced with a complex process of adding a new, higher layer of control that appropriately inhibited the old behaviors and expressed itself instead. This typically involves debugging the interfaces between a large number of behaviors — each behavior in the layer that was most recently controlling the robot with all the behaviors in the new layer of control — rather than just the one that is being adjusted.

BOD vs. OOD OOD tends to be useful for passive, reactive systems, but not to be a good methodology for designing systems that are actively internally motivated. The addition BOD provides over OOD is the reactive plan component — this allows the expression of motivation as part of the organization of behavior. BOD applies techniques for building plans and decomposing behaviors that are analogous to, but not exactly the same as the OOD methodologies for designing object hierarchies. A very brief summary of what would differ can be gathered from this statement: the behaviors in BOD are not hierarchical, the reactive plans are.

BOD vs Agent Oriented Design (AOD) AOD assumes that every element is an agent, with intentions and fully encapsulated state. BOD allows different behaviors to have access to each other's state, and worries about intentions and planning on a global level, between behaviors, not within them. AOD is actually more analogous to standard BBAI than to BOD.

9 Example of BOD: design of a mind for an interactive tutoring system

This is two projects in progress. One is a dialog agent designed to tutor students in basic electricity. This is an ongoing project in Edinburgh, my part is to determine how much of dialog management should be reactive, and how it should be implemented. The other project is one of design: using the same principles to "design a mind" in response to a CFP by Aaron Sloman. I intend to discuss using the BOD methodology to design the various elements one would expect in a humanoid agent, and describe how they can interface. Elements include motivation (affect), episodic memory, semantic memory, and procedural memory. Desired behaviors include dialog and social learning. I don't expect to complete an entire system for this thesis, but rather to present the various sections. This section is dedicated to overall design, with the working demonstration being the dialog manager for the tutoring system.

As mentioned in Section 7, there may also be some work on modeling deliberate vs. automatic behavior. This would be done along the lines suggested by Norman & Shallice (1986), whose proposal for an architecture for automatic action selection remarkably similar to my own.

10 Examples of BOD with specialized learning within the dialog context

This chapter is intended to integrate learning into the system described in the previous section. However, since it is not clear that the various pieces described will operate reliably or in real time, they are presented as separate research projects. The first two address open problems in the field of natural language.

The first project is to use a learned lexicon to understand metaphor. This project is designed to show the interaction between a behavior generated by specialized unsupervised learning and the action selection mechanism. It involves combining recent technology which essentially automatically generates a semantic net by operating unsupervised learning (with some stringent biases) across large corpora of text. This network as the state of a lexical semantic "behavior", where activation levels of words in this network indicate the current understanding of the agent. Using a strategy similar to that developed by Ullman and Chapman to describe visual search, a reactive plan will then parse a sentence such as "Since the holidays, the scale has been my worst enemy." to try to determine its gist (weight.) Results from this project would be particularly significant if they can be used to explain reaction time results in metaphor work similar to those that have been demonstrated in vision. For example, if a relationship can be found between the RT and the number of words that need to be visited by the algorithm before a “solution” is found.

The second project is to extend the lexicon through experience. This would show the interaction between explicit / fast learning and implicit / slow learning. The semantic net would be represented in an LSA table. Explicit rules would detect a novel word, or a novel usage of a word. This word or word instance would then be added to the table, and future unsupervised learning would treat it as another node of the network. In the case where this is the addition of a new word sense, the original / dominant word sense would be able to migrate closer to its ordinary meaning since the new sense would serve as a separate attractor for the auxiliary meaning. The standard example of word sense disambiguation is a river bank vs. financial bank. Determining the number of word senses is an unsolved problem in large corpus semantics.

A third project that would be desirable would be to demonstrate the acquisition of grammatical or discourse structures, or in other words, to automatically acquire reactive plans. Dawkins published some work on learning hierarchies of animal behavior in 1975 based on clustering algorithms, and current clustering algorithms far excel those of the time. I'm uncertain that time will permit this as a project (or, if it does, whether I'd rather do this in language or in gesture imitation, see below.) But since this is also an active area of research, the recent results would at least be discussed if not demonstrated.

11 Non-Linguistic Examples: Social Learning in Non-Human Primates

This chapter is optional, I may prefer to do more work on language, particularly if a single large dialog system look plausible. However, this stuff is trendier and better looking, and I've always preferred working on animal rather than human intelligence. This chapter might also have no demonstrations, but exist anyway as a review of biological and/or neurological implications. I have already found at least two local research groups doing highly relevant work in this area. Earl Miller's group in BCS is working on behavior arbitration via the prefrontal cortex, I've already spoken to him and he's very interested in my work. I just heard Ronald Rensink talk about his work on how attention moves visual representation from parallel distributed but atemporal substrates into “bound” and temporally / episodically useful information, he is at CBR. I've also been told that Elizabeth Spelke, also of BCS, is very interested on how language and/or other attentional mechanisms overcomes the massive modularity of infant behavior and competencies. Feedback from these groups plus the information I already know could lead to a very interesting review, and possibly some modeling.

One possibility is to make this chapter analogous to the previous one, for example starting with a discussion of the categorization learning literature as analogous to lexicon work, and imitation work as analogous to grammar learning.

I have already done some work on a cognitive model of primate imitation with Andrew Whiten, but this work is currently on hold.

12 Conclusions

See Section 1.

12.1 Evidence from the Large, Fancy System(s) — or When Does it End?

My feeling is that the BOD methodology is already adequately described, and the literature research support for it is also already completed. I have a journal article in press on the the analysis. The real question then is how much of a system is necessary in order for me to finish? As I stated earlier, there's no good criteria on this, because it's not really a very scientific approach. So I hope that the answer is just "something pretty publishable," which I expect to get done this summer. I hope to have at least one article on using BOD to construct a dialog system, a couple of short papers on advances in computational linguistics and unsupervised natural language learning using BOD, and possibly a paper on using BOD to model imitation learning in non-linguistic primates. I think that if 3 of those 4 papers are good enough to get published, I should have enough material to finish the thesis.

13 Timeline

All the scheduling and writing information has been updated on line. Please see http://www.ai.mit.edu/ joanna/proposal/proposal.html

```
Timeline

   February 2000
 S  M Tu  W Th  F  S
       1  2  3  4  5        hack on dialog system
 6  7  8  9 10 11 12        work on proposal
13 14 15 16 17 18 19        proposal writing + format Tyr for SAB
20 21 22 23 24 25 26        hack on dialog / metaphor
27 28 29                    AISB design of mind / dialog system (+ ethics paper)

   March 2000
 S  M Tu  W Th  F  S
          1  2  3  4        (set up MIT office, more proposal)
 5  6  7  8  9 10 11
12 13 14 15 16 17 18        hack on metaphor / word sense, BOD in JAM
19 20 21 22 23 24 25        ATAL paper / chapter on reactive plans, POSH
```

```
26 27 28 29 30 31          hack on proposal, JAM, AISB slides

  April 2000
 S  M Tu  W Th  F  S
                   1
 2  3  4  5  6  7  8       evol of lang (paris)
 9 10 11 12 13 14 15       hack on tutoring in Edin
16 17 18 19 20 21 22       AISB (Birmingham)
23 24 25 26 27 28 29       hack on metaphor / proposal
30

   May 2000
 S  M Tu  W Th  F  S
    1  2  3  4  5  6       vr journal paper rewrite (with Kris)
 7  8  9 10 11 12 13       ATAL rewrite 8-11 pt
14 15 16 17 18 19 20
21 22 23 24 25 26 27       finish metaphor
28 29 30 31                28 vr j paper due, 30-1 Emernet abstract

   June 2000
 S  M Tu  W Th  F  S
             1  2  3       word sense disambiguation
 4  5  6  7  8  9 10       6-7 NCPW6 Abstract?
11 12 13 14 15 16 17       grammar or discourse learning
18 19 20 21 22 23 24
25 26 27 28 29 30

   July 2000
 S  M Tu  W Th  F  S
                   1
 2  3  4  5  6  7  8       7-9 ATAL (boston) / robocup rescue on 8th
 9 10 11 12 13 14 15       imitation learning / VR agents?
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31

   August 2000
 S  M Tu  W Th  F  S
       1  2  3  4  5
 6  7  8  9 10 11 12       8-9 EmerNet (Durham)
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

```
Fall Term               minor, area exam, complete draft of thesis.
                          September: SAB (Paris), NCPW6 (Liege),
                                     humanoid robots (Boston)
                          November: vrj final due.
Spring Term             minor, papers and projects, hunt jobs.
                          CogSci 2001 in Edin.
```

14 Hacking Schedule

- rewrite dialog stuff in something other than SRI's PRS-CL, which sucks. Probably JAM, a java derivative of UM-PRS (Michigan's version, which is dominant and in C++, but maintenance is falling behind.) — feb - april
- the analogy project — by end of May
- more on tutoring / static dialog system (coordinates with Edin. visit) – around 14 april, perhaps late summer
- the word sense disambiguation project — by end of June.
- grammar learning project — May, maybe June – depends on time of analogy and disambig. projects
- imitation project / social learning July, August, maybe Fall Term. – unless replaced by "complete" dialog system
- remainder of Fall, Spring driven by papers, projects spinning off from above. Mostly intend to be writing up.

15 Dissertation source materials

1. Introduction — (Mostly this document, also web,SAB/Tyrrell,JETAI) - source papers essentially complete
2. the place of learning (AISB96 paper+old proposal) - source papers complete
3. fittest architectures: evidence from literature (JETAI paper) - source paper complete
4. posh action selection (prioritized slip-stack hierarchies) (ATAL 200 paper) - done by end of may (draft complete)
5. behavior oriented design (AISB00-mind paper, this proposal, complete) rule list may still expand some.
6. design of a dialog system (draft complete in AISB00-mind), will extend with HCRC tutoring dialog system and/or my dialog stuff in fall.
7. specialized learning in a dialog system (hopefully cogsci 2001, research by July)
8. specialized learning in primates (research this summer (unless extending dialog system), maybe fall. Some draft work on paper model already done)
9. Conclusions

An alternative ordering for the last three chapters might unify the dialog and dialog learning chapter, and have a following chapter on designing "minds", based on the paper with Glasspool, and the biological plausibility paper if it is delivered at Emernet.

16 Collection of articles serving as dissertation source

To complete this proposal, I include a packet of the articles listed above as already complete and providing source. If you don't have the packet, you should look for this section on line as it has links.

- I omit the articles for the introduction; that is fairly well outlined above. Though note that the ATAL 2000 paper listed below does a good job of differentiating between an architecture, a methodology and a toolset, and talking about the contributions of each.
- For "The Place of Learning" I include "The Design of Learning for an Artifact" which appeared in the proceedings of an AISB 96 symposium on learning in animals and robots, and a draft article "Specialized Learning and the Design of Intelligent Agents." This article is weak because it doesn't contain a final example, which should be the research proposed for this Spring and Summer. It was written in the Summer of 1997 when I was here in the AI lab.
- For "The Fittest Architectures" I have a very recent (apparently nearly in press) review article for JETAI "Cross-Paradigm Analysis of Autonomous Agent Architecture." Section 2 of this article is nearly verbatim from my MPhil's review section, so would not appear in the PhD.
- As a draft of the POSH chapter, I include "Architectures and Idioms: Making Progress in Agent Design", which has been accepted for publication in ATAL 2000. It talks about what reactive planning is, about POSH, and about implementing POSH both in Edmund and in other, more standard architectures.
- As a draft for the BOD chapter I have (from AISB 2000) "Making Modularity Work: Combining Memory Systems and Intelligent Processes in a Dialog Agent". This talks about BOD in detail, it also has an introduction to the POSH stuff, and also a rough sketch of the dialog work.

Bibliography

Benson, S. (1996), Learning Action Models for Reactive Autonomous Agents, PhD thesis, Stanford University. Department of Computer Science.

Blumberg, B. M. (1996), Old Tricks, New Dogs: Ethology and Interactive Creatures, PhD thesis, MIT. Media Laboratory, Learning and Common Sense Section.

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* 'Proceedings of the 1991 International Joint Conference on Artificial Intelligence', pp. 569–595.

Bryson, J. (1999*a*), Creativity by Design: A Behaviour-Based Approach to Creating Creative Play, *in* F. Nack, ed., 'AISB'99 Symposium on Creativity in Entertainment and Visual Art', Sussex, pp. 9–16.

Bryson, J. (1999*b*), Hierarchy and Sequence vs. Full Parallelism in Action Selection, *in* D. Ballin, ed., 'Intelligent Virtual Agents 2'.

Bryson, J. (2000), The Integration and control of behaviour: Insights from neuroscience and AI, *in* A. Sloman, ed., 'AISB'00 Symposium on Designing a Functioning Mind'.

Bryson, J. (in preperation), Architectures and Idioms: Making Progress in Agent Design, *in* 'The Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL2000)'. to be presented July 2000.

Bryson, J. (in press), 'Cross-Paradigm Analysis of Autonomous Agent Architecture', *Journal of Experimental and Theoretical Artificial Intelligence*.

Bryson, J. & McGonigle, B. (1998), Agent Architecture as Object Oriented Design, *in* M. P. Singh, A. S. Rao & M. J. Wooldridge, eds, 'The Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL97)', Springer-Verlag.

Chapman, D. (1987), 'Planning for conjunctive goals', *Artificial Intelligence* **32**, 333–378.

Coad, P., North, D. & Mayfield, M. (1997), *Object Models: Strategies, Patterns and Applications*, 2nd edn, Prentice Hall.

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

Fodor, J. A. (1983), *The Modularity of Mind*, Bradford Books, MIT Press, Cambridge, MA.

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

Hinton, G. E. & Nowlan, S. J. (1987), 'How Learning Can Guide Evolution', *Complex Systems* **1**, 495–502.

Kline, C. & Blumberg, B. (1999), The Art and Science of Synthetic Character Design, *in* 'AISB'99 Symposium on AI and Creativity in Entertainment and Visual Art'.

Laird, J. E. & Rosenbloom, P. S. (1994), The Evolution of the Soar Cognitive Architecture, Technical Report CSE-TR-219-94, Department of EE & CS, University of Michigan, Ann Arbor. also in *Mind Matters*, Steier and Mitchell, eds.

Maes, P. (1991), 'The agent network architecture (ANA)', *SIGART Bulletin* **2**(4), 115–120.

Matarić, M. J. (1997), 'Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior', *Journal of Experimental & Theoretical Artificial Intelligence* **9**(2/3), 323–336.

Nilsson, N. (1984), Shakey the Robot, Technical note 323, SRI International, Menlo Park, California.

Norman, D. A. & Shallice, T. (1986), Attention to Action: Willed and Automatic Control of Behavior, *in* R.Davidson, G. Schwartz & D. Shapiro, eds, 'Consciousness and Self Regulation: Advances in Research and Theory', Vol. 4, Plenum, New York, pp. 1–18.

Reilly, W. S. N. (1996), Believable Social and Emotional Agents, Technical Report CMU-CS-96-138, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Ph.D. thesis.

Schoppers, M. (1995), 'The use of dynamics in an intelligent controller for a space fairling rescue robot', *Artificial Intelligence* **73**, 175–230.

Sloman, A. (2000), Models of Models of Mind, *in* A. Sloman, ed., 'AISB'00 Symposium on Designing a Functioning Mind'.

Stein, L. A. (1994), 'Imagination and Situated Cognition', *Journal of Experimental and Theoretical Artificial Intelligence* **6**, 363–407.

Thórisson, K. R. (1999), 'A Mind Model for Multimodal Communicative Creatures & Humanoids', *International Journal of Applied Artificial Intelligence* **13**(4/5), 519–538.

Thórisson, K. R. & Bryson, J. (in preperation), 'Flexible Behavior-Based Planning for Multimodal Characters Capable of Task-Oriented Dialogue and Action'.

Turney, P., Whitley, D. & Anderson, R. (1996), 'Evolution, Learning, and Instinct: 100 Years of the Baldwin Effect', *Evolutionary Computation*. special issue.

Tyrrell, T. (1993), Computational Mechanisms for Action Selection, PhD thesis, University of Edinburgh. Centre for Cognitive Science.

Weld, D. S. (in press), 'Recent Advances in AI Planning', *AI Magazine*.

Wooldridge, M. & Jennings, N. R. (1995), 'Intelligent Agents: Theory and Practice', *Knowledge Engineering Review*.