Embodying a Cognitive Model in a Mobile Robot

D. Paul Benjamin[a], Damian Lyons[b], Deryle Lonsdale[c]

[a]Pace University Computer Science Department, 1 Pace Plaza, New York, New York 10038;
[b]Fordham University Department of Computer & Information Science 340 JMH, Fordham University, 441 E. Fordham Rd., Bronx, New York 10458;
[c]Brigham Young University Department of Linguistics and English Language Provo, Utah 84602

ABSTRACT

The ADAPT project is a collaboration of researchers in robotics, linguistics and artificial intelligence at three universities to create a cognitive architecture specifically designed to be embodied in a mobile robot. There are major respects in which existing cognitive architectures are inadequate for robot cognition. In particular, they lack support for true concurrency and for active perception. ADAPT addresses these deficiencies by modeling the world as a network of concurrent schemas, and modeling perception as problem solving. Schemas are represented using the RS (Robot Schemas) language, and are activated by spreading activation. RS provides a powerful language for distributed control of concurrent processes. Also, The formal semantics of RS provides the basis for the semantics of ADAPT's use of natural language. We have implemented the RS language in Soar, a mature cognitive architecture originally developed at CMU and used at a number of universities and companies. Soar's subgoaling and learning capabilities enable ADAPT to manage the complexity of its environment and to learn new schemas from experience. We describe the issues faced in developing an embodied cognitive architecture, and our implementation choices.

Keywords: Mobile robotics, cognitive architecture, predictive vision, learning, robot schemas, ADAPT

1.INTRODUCTION: EMBODIED COGNITION, REFORMULATION AND ROBOTICS

There is a growing body of scientific work based on the belief that the mind must be understood in terms of controlling a physical body acting in the real world. This belief, often referred to as *embodied cognition*, strongly contrasts with the belief that the mind is an abstract computer. The abstract computational model of mind has attained some notable successes in very specific tasks such as chess, but has not done as well in robotics. Robots have a great deal of difficulty understanding their environment. Cognitive science has constructed cognitive architectures that model reasoning and learning in individual tasks and that reason about spatial and temporal relationships in simulated domains. We believe that it is time to embed a cognitive model in the physical world and test these reasoning and learning mechanisms when faced with the full complexity of the real world

The ADAPT project (**A**daptive **D**ynamics and **A**ctive **P**erception for **T**hought) is a collaboration of three university research groups at Pace University, Brigham Young University, and Fordham University to produce a robot cognitive architecture that integrates the structures designed by cognitive scientists with those developed by robotics researchers for real-time perception and control. Our goal is to create a new kind of robot architecture capable of robust behavior in unstructured environments, exhibiting problem solving and planning skills, learning from experience, novel methods of perception, comprehension of natural language and speech generation.

The current generation of behavior-based robots is programmed directly for each task. The programs are written in a way that uses as few built-in cognitive assumptions as possible, and as much sensory information as possible. The lack of cognitive assumptions gives them a certain robustness and generality in dealing with unstructured environments. However it is proving a challenge to extend the competence of such systems beyond navigation and some simple tasks [46]. Complex tasks that involve reasoning about spatial and temporal relationships require robots to possess more advanced mechanisms for planning, reasoning, learning and representation.

One approach to this issue is to equip a robot with a behavior-based "reactive" module coupled to a "deliberative" module that engages in cognitive reasoning – these are called hybrid reactive/deliberative systems [2]. Many such systems consist of a symbolic planner, or planner and scheduler, as the deliberative component. The planner formulates long-term and more abstract tasks, while the behavior-based system carried out the steps in the task in a robust fashion [2]. For example, Lyons et al. [35,36] describe a hybrid system based on the concept of iterative and forced relaxation of assumptions about the environment. The deliberative module generates and maintains a behavior-based system that is as robust as possible, given the tradeoff between the time available to the planner and any previously sensed assumption failures. The reactive system is implemented using the port-automata based RS model [37]. The hybrid approach endows a robot with a more sophisticated planning ability than either deliberative planning or reactive behavior provides alone; however, it does not address either learning or representation issues.

Our approach is fundamentally different from other hybrid architectures, which typically attempt to build a comprehensive system by connecting modules for each different capability: learning, vision, natural language, etc. Instead, we are building a *complete cognitive robotic architecture* by merging RS [37,38], which provides a model for building and reasoning about sensory-motor schemas, with Soar [21], a cognitive architecture that is under development at a number of universities. RS possesses a sophisticated formal language for reasoning about networks of port automata and has been successfully applied to robot planning [36]. Soar is a unified cognitive architecture [45] that has been successfully applied to a wide range of tasks including tactical air warfare [50].

This cognitive robotic architecture is currently being implemented and tested on Pioneer P2 robots (http://robots.activmedia.com/) in the Pace University Robotics Lab (http://csis.pace.edu/robotlab) and the Fordham University Robotics Lab (http://www.cis.fordham.edu/rcvlab). Three publications describing our initial work on ADAPT are [4,5,6].

1.1. Our Approach to Integrating Cognition with Robotics

Our experience in industrial and academic robotics research has taught us a number of lessons about the nature of robotics that are shared by many in the field, including:

- All robotic activity is sensorimotor, encompassing both perceptual and motor aspects,
- Perception is active, goal-directed and task-dependent, and
- Control is distributed and concurrent, with a high degree of parallelism.

Even "pure" vision is sensorimotor, as it involves selecting a target, framing it and tracking it, which depends on the task and goals of the robot. While carrying out vision tasks, the robot cannot stop attending to its other sensors; in addition, the robot may be moving. Thus, robot programs typically consist of many concurrent routines. Only at the highest levels of abstraction can these programs be described as composed of sequential actions. At these high levels of the abstraction hierarchy, the actions are composed sequentially and search is necessary to identify the choice of action at each step to make progress towards a goal. In the lower levels of the abstraction hierarchy, the actions are composed concurrently and search is necessary to identify the set of actions to compose to obtain a desired behavior.

Integrating Soar and RS is a straightforward way to try to build an architecture that can handle the entire abstraction hierarchy smoothly.

One of the most unique and important aspects of our architecture is its treatment of perception and language and their relationship to knowledge representation. Our view of perception is that it is an "active" process [15] that is goal-directed and task-dependent, i.e. it is a cognitive problem-solving process rather than a peripheral activity separate from cognitive processing. Furthermore, we view perception as intimately linked with the formation and modification of symbolic representations, so that perception's main purpose is to identify, build and modify representational structures that make the robot's goals easier to achieve. In this view, the robot solves problems primarily by searching among different ways of perceiving the world and the task. This is in contrast to the usual approach of searching among sequences of actions in one or a few fixed representations. Each way of perceiving the world and task leads to a distinct symbolic formulation. This process of *reformulation* is based on algebraic linguistic structures that are the formal basis of the integration of Soar and RS.

In the next two sections, we describe the various components of ADAPT. The subsequent section describes how reformulation relates to the integration of RS and Soar.

1.2. RS and Soar

RS is a formal model of robot computation that is based on the semantics of networks of port automata [57]. A port automaton (PA) is a finite-state automaton equipped with a set of synchronous communication ports [38]. Formally we can write a port automaton P as:

$P = (Q, L, X \delta, \beta, \tau)$ where

Q is the set of states

L is the set of ports

$X = (X_i \mid i \in L)$ is the event alphabet for each port

Let $XL = \{ (i, X_i) \mid i \in L \}$ i.e., a disjoint union of L and X

$\delta : Q \times XL \rightarrow 2^Q$ is the transition function

$\beta = (\beta_i \mid i \in L)$ $\beta_i : Q \rightarrow X_i$ is the output map for port i

$\tau \in 2^Q$ is the set of start states

RS introduces a vocabulary to specify networks of port automata. A network of processes is typically built to capture a specific robot sensorimotor skill or behavior: sensory processes linked to motor processes by data communication channels and sequenced using *process composition operations*.

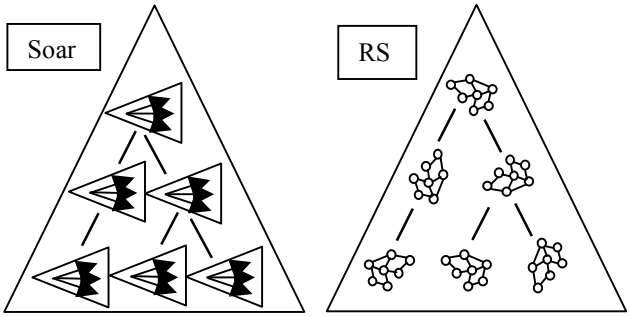
RS process composition operations are similar to the well-known CSP algebraic process model of [53]. However, unlike CSP, in RS the notation can be seen as simply a shortcut for specifying automata; a process is a port automaton, and a process composition operation is two automata connected in a specific way. Composition operations include sequential, conditional and disabling compositions [37,38]. To analyze a network of processes, it is necessary to calculate how that network *changes* as time progresses and processes terminate and/or are created. This is the process-level equivalent of the PA transition function, combined with the axioms that define port-to-port communication. This *Process Transition function* can be used to analyze the behavior of RS networks [39]. RS is implemented in Soar.

Soar [45] is a cognitive architecture originally developed at CMU and undergoing continuing development at a number of locations, including the University of Michigan and the Information Sciences Institute. Knowledge in Soar is represented as operators, which are organized into problem spaces. Each problem space contains the operators relevant to some aspect of the system's environment. In our system, some problem spaces contain operators describing the actions of the robot for particular tasks and subtasks. Other problem spaces contain operators governing the vision system, including operators about how to control the camera, operators for selecting software to process the visual data, and operators for the creation and modification of local coordinate systems in the visual data.

The basic problem-solving mechanism in Soar is *universal subgoaling*: every time there is choice of two or more operators, Soar creates a subgoal of deciding which to select, and brings the entire knowledge of the system to bear on solving this subgoal by selecting a problem space and beginning to search. This search can encounter situations in which two or more operators can fire, which in turn causes subgoals to be created, etc. When an operator is successfully chosen, the corresponding subgoal has been solved and the entire solution process is summarized in a single rule, called a chunk, which contains the general conditions necessary for that operator to be chosen. This rule is added to the system's rule set, so that in similar future situations the search can be avoided. In this way, Soar learns. The chunking mechanism has proved successful on a variety of tasks [40], and is well suited to learning schemas of port automata.

The Soar publications extensively document how this learning method speeds up the system's response time in a manner that accurately models the speedup of human subjects on the same tasks [20]. In addition, the Soar research community is implementing a wide range of aspects of cognition in Soar, including natural language [31, 33], concept learning [42,59] and emotion [41].

Soar manipulates a hierarchy of problem spaces, and the formal semantics of RS consists of a hierarchy of port automata. We have merged these architectures in a straightforward way by implementing each RS schema as a Soar problem space. This is done by specifying the state transitions of each schema's port automaton in Soar.



Merging RS and Soar in this way combines their strengths. The strengths of RS include its formal mechanism for combining sensing and motion, its ability to reason about the temporal behavior of schemas, its combination of deliberative planning and reactive behavior, and its real-time parallel scheduling and execution environment. Its weaknesses are the lack of a mechanism for autonomous formation of sensors or actuators, and the lack of a model for implementation of cognitive abilities such as learning and language.

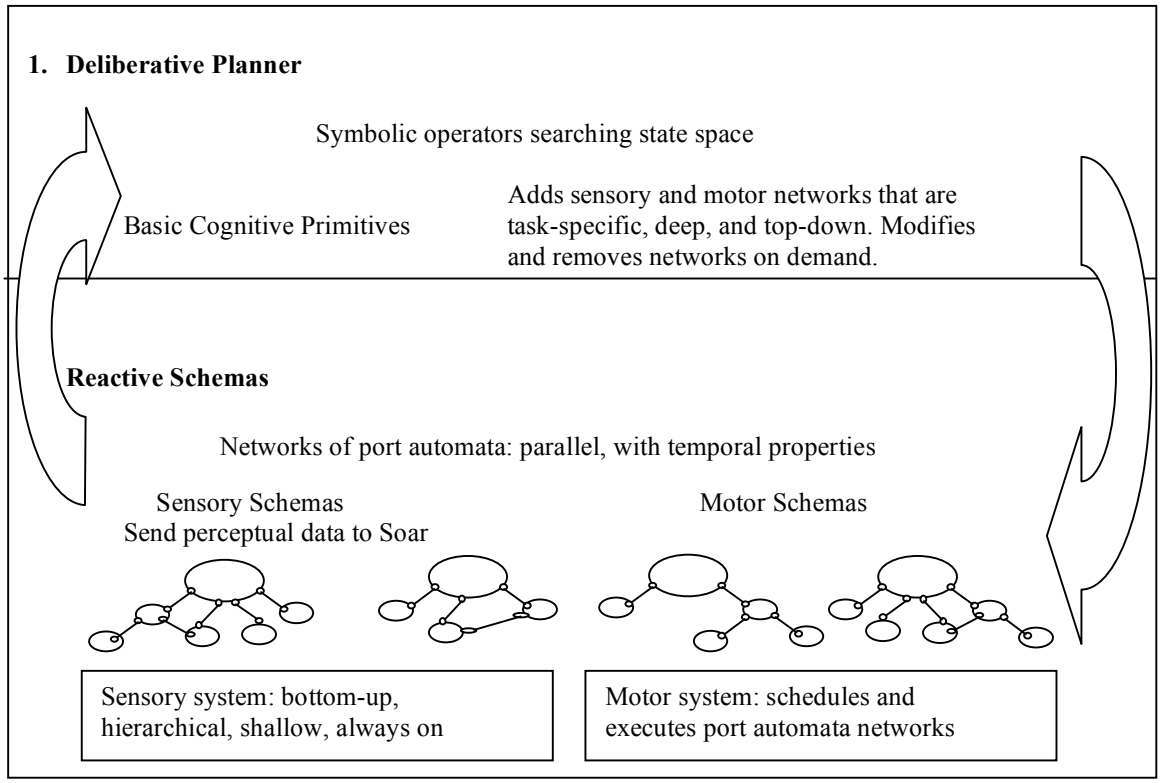
Soar provides an integrated cognitive model with a full range of cognitive abilities, including perception, deliberative planning, reaction, natural language, learning, and emotion. But Soar lacks parallelism and a temporal mechanism, which severely hampers its usefulness in robotics.

By integrating RS and Soar, we are creating an architecture that will:

- process a wide range of modes of perceptual input,
- provide a complete range of cognitive abilities, including language and learning,
- reason about time and resources,
- implement parallel, reactive behaviors, and
- learn new high-level sensors and behaviors.

Behaviors are specified and implemented in RS, which provides an execution environment for automata networks. The basic sensory system is always on and is hierarchical, shallow and bottom-up. The deliberative module can instantiate automata networks on demand, and can add networks to the sensory system that are task-specific, deep, and top-down.

The deliberative component is implemented in Soar. It receives information sent by the executing reactive network, and can instantiate or stop networks. Soar reasons about the automata networks that are used to represent the environment and the task, and generates abstract plans that are refined into automata networks, creating new networks as necessary.



Soar's learning mechanism can form net automata networks by chunking, as well as learn new plan components and learn search control knowledge to improve planning and reaction time.

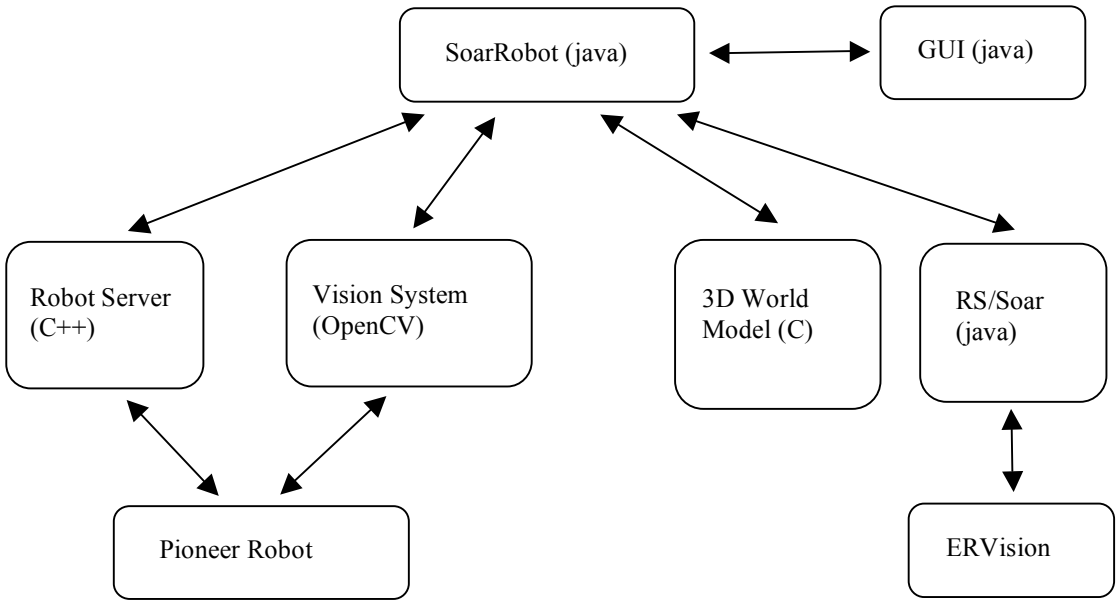
1.3. The World Model and its use in Perception and Planning

Another distinctive feature of ADAPT is that it uses a powerful multimedia world model. In the current implementation, ADAPT's world model is the Ogre3D open source gaming platform (http://www.ogre3d.org). Ogre gives the robot the ability to create a detailed and dynamic virtual model of its environment, by providing sophisticated graphics and rendering capabilities together with a physics engine based on ODE (Open Dynamics Engine). Ogre is capable of modeling a wide variety of dynamic environments, and also of modeling other agents moving and acting in those environments.

The robot uses this world model in a fundamentally different way than is typical in other work. ADAPT's world model is connected only to Soar, not directly to the external world, so that sensory data must be processed by Soar and modeled in Ogre. This reflects our belief that perception is an active process; perception is *not* done by peripheral processes that interpret the sensory data and create entities and relationships in the world model. Instead, ADAPT makes explicit decisions about how to process the sensory data, e.g. what visual filter to apply to vision data, and explicitly models the results in Ogre. In this way, perception becomes a problem-solving process, capable of drawing on all the knowledge of the system and permitting Soar's learning capability to be applied to perception.

Ogre embodies the graphical and dynamic aspects of ADAPT's world model. Soar contains the symbolic part of the world model, which consists of "annotations" describing the entities, relationships and activities in Ogre. For example, when Soar recognizes a person sitting in a chair, it will construct virtual copies of the chair and the person in Ogre and create symbolic structures in Soar's working memory that point to them, as well as a symbol structure for the relationship of sitting. Ogre serves as the model that interprets the symbols in Soar's working memory. The relationships between the Soar and Ogre parts of the world model are updated automatically each Soar cycle.

The overall structure of ADAPT is shown below.



The above diagram shows the major components of ADAPT. The lower left part of the diagram shows the two components that run on the PC that is onboard the Pioneer Robot. These are the robot server software that provides the basic interface to the robot, and the bottom-up component of the Vision System, which provides basic visual data including stereo disparity (distance information), segments (color blobs), and optical flow (motion information). The

robot server is a C++ program that interfaces with the robot's hardware. The server program listens on a socket for a call from SoarRobot, then packs all the robot's data (including the sonar data) into C structs that are sent back to SoarRobot. While the server is waiting, the Vision System runs on the robot's PC.

The lower right part of the diagram shows the components that run on the base PC. These include the 3D world model, which is currently the Ogre3D gaming engine (www.ogre3d.org), and Soar, which contains an implementation of RS schemas. Soar also can call the ERVision object recognition software.

The GUI is currently the Soar Debugger, which provides a nice interface.

The entire system is held together by the SoarRobot program, which is written in Java. Most of the other components have Java interfaces, except for Ogre3D, which has a C language interface that is wrapped in Java. SoarRobot coordinates the interaction of all components and the user (via the GUI), and runs on the base PC.

The basic loop of SoarRobot is:

1 - check Soar's output link to see if there are any commands, which may be either motion commands for the robot or modeling commands for the World Model,

2 - blend the motion commands that are to be sent to the robot,

3 - send all robot commands both to the robot and to the virtual robot in the World Model,

4 - send all other commands to the World Model,

5 – periodically (every tenth of a second) fetch data from the robot to be put into Soar's working memory,

6 - periodically fetch data from the Vision System, compare it to visual data from the World Model, and put any significant differences into Soar's working memory.

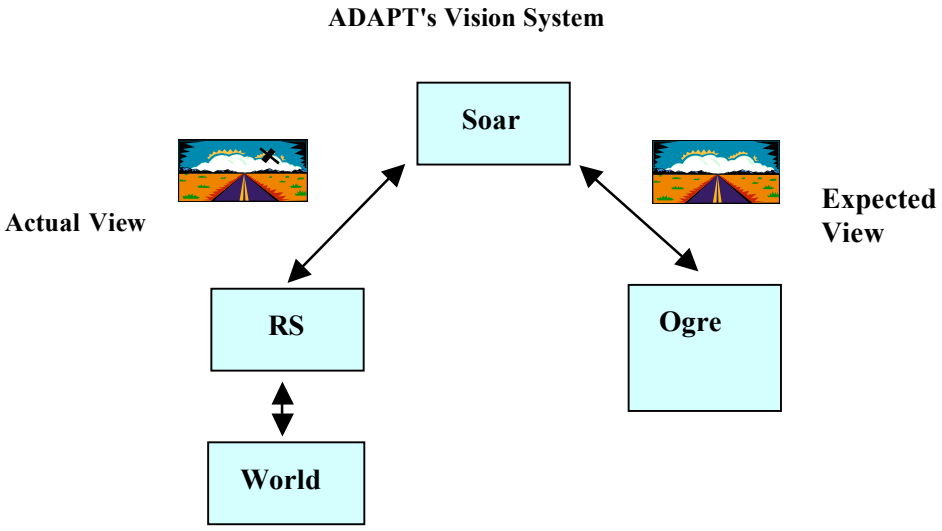
This world model also permits ADAPT to visualize and explore possibilities by firing Soar operators that create structures in the world model that do not correspond to sensory data, i.e. it can "imagine" possibilities. Using the physics plugin, the robot can examine how the world might evolve in the near future. ADAPT's planner will use this visualization capability in the same way that NL-Soar understands the world using "comprehension through generation" [18]. In this approach to comprehension, NL-Soar understands an utterance by searching for ways to generate it. ADAPT will comprehend its environment by searching among ways to recreate it. Comprehension through generation is a slow process, and depends upon the speedup of chunking to be practical. Evaluating the success of this approach to modeling the world is one of the central goals of the project.

As an illustration of the use of this world model, let us consider ADAPT's novel approach to visual comprehension of its environment. ADAPT's vision system consists of two main components, a bottom-up component that is always on, and a top-down goal-directed component. The bottom-up component segments the visual data from the robot's two framegrabbers; this component runs on the robot's onboard computer. The output of this component is a set of "blobs" that are transmitted to an offboard PC which is running Soar. A description of these blobs is placed into Soar's working memory during the initial phase of each execution cycle of Soar.

The graphics "camera" in Ogre produces the view that the virtual copy of the robot "sees" in the virtual environment. The output of this camera is also segmented and placed into Soar's working memory. Soar operators test for significant differences between the expected view and the actual view, e.g. a large new blob. Any significant difference causes a Soar operator to be proposed to look at this blob and try to recognize it. If this operator is selected (if there is nothing more important to do at the moment) then Soar will turn its cameras towards this blob, and then call its recognition software to process it.

Once the object is recognized, a virtual copy is created in Ogre. The object does not need to be recognized again; as long as the blobs from the object match the expected blobs from Ogre, ADAPT assumes it is the same object.

Recognition is thus an explicitly goal-directed process that is much cheaper than always recognizing everything in the environment. Furthermore, this approach to recognition opens the possibility of learning recognition strategies.



1.3.1. Reformulation and Emergent Properties

ADAPT's components are not merely connected but integrated, and this integration is based on a formal relationship between the basic hierarchical structures of Soar and RS. ADAPT's approach to problem solving differs significantly from the traditional planning approach. ADAPT searches for a set of local symmetries and invariants that can be used to decompose the robot's environment and task into easily analyzed pieces [13].

Each formulation of knowledge possesses different computational properties. Given a particular task in a particular environment, the robot must choose a formulation whose computational requirements are well suited to the task. A good choice makes the task easier, in a manner that is very similar to that encountered in solving basic calculus problems, where for example, using a Cartesian coordinate system makes some problems easy to solve, while using polar coordinates makes others easy.

Of special importance is finding a decomposition of the robot/environment system into components whose local properties compose to determine global properties. Such decompositions reduce problem-solving complexity by permitting the robot to solve a set of small, local problems; the global properties and solutions emerge from the interaction of the local properties and solutions [7, 8, 9]. The robot must identify these components and use them to formulate the task in order to solve it efficiently.

The mathematics of formal languages and grammars, and especially that of algebraic linguistics [17], [22], [47], provides formal tools for analysis of this language of behaviors and synthesis of different grammars. In particular, it provides theorems and algorithms for decomposing this language into a hierarchy of sublanguages. The components of this hierarchy are languages of varying granularities, each of which perceives and acts on a subset of the environment. Each component is characterized in terms of its symmetries and invariants.

This decomposition hierarchy underlies perception, problem solving and natural language, as the local symmetries perceived in the environment correspond to the preconditions and invariants of the components used to solve problems. In [14] we show how this structure appears in robot motion planning, and how it can be exploited to reduce the cost of planning. Perception exhibits such a structure, too, as Barnsley [3] has shown that a self-similar structure exists in images and has developed fractal compression methods that are commercially viable competitors in the areas of image processing and communications. Similarly, structure and meaning in natural language are largely compositional.

Learning this one fundamental structure is superior to a patchwork approach of gluing together learning processes that are separately developed, as is typical in hybrid architectures. It is certainly possible to glue together a method of learning visual object models, a method of learning problem-solving heuristics and a method of learning natural

language. But such an approach tends to eliminate useful interactions between these processes, e.g. that learning a new way of solving a class of problems could improve perception or communication. We feel that such interactions are not only common, but are essential in the quest to develop a robot with human-level proficiency in an unstructured environment.

Program structures for all the processes of an intelligent agent should embody this hierarchy in a straightforward way. Soar is an implementation of Rosenbloom and Newell's Problem Space Computational Model [49], which models this neighborhood structure for cognitive processes; RS's hierarchy of port automata models it for sensory and motor processes. An architecture integrating Soar and RS offers a coherent robot control system based on this structure. Indeed, *universal subgoaling reflects the self-similarity of problem solving*. This is the main reason for the choice of Soar as the software for the deliberative component of the cognitive architecture - the self-similar structure of Soar's problem solving matches the self-similar structure of the environment.

Our decomposition algorithm finds a task's hierarchy of local neighborhoods, and reformulates the task in terms of local coordinate systems for these neighborhoods. The relevant properties for system formulation are those properties that hold locally everywhere within the neighborhoods. Simple programs can measure these local properties and maneuver within local neighborhoods. Under certain conditions, the local neighborhood structure becomes a substitution tiling [54]. In these cases, self-similarity of the neighborhoods permits the simple programs to be effective at several scales, creating a hierarchical structure that is simple and local at each scale, but capable of generating very complex emergent global behaviors. Fortunately, substitution tilings appear to be ubiquitous in the physical world [54].

Local coordinate systems define the perceptual spaces for the robot. The axes in each coordinate system correspond to a set of features for describing the state space. Associated with the local symmetries of the local coordinate system are invariant subspaces of states (states on which particular combinations of features are invariant.) Motion in the state space decomposes into motion within invariant subspaces and motion normal to those subspaces. This decomposition reduces the complexity of motion planning, and also creates the goal for the perceptual system: the robot must perceive the visual symmetries and monitor the invariants of the motions. The features determined in this way are the semantic components that form the basis of communication [10, 11].

The implications of this structure for planning and problem solving are clear: instead of using combinatorial search in the space of possible solutions, a robot can search for a self-similar local neighborhood structure in the constraints of the task and the environment. When such a structure exists, the robot can exploit it to solve large, complex planning problems in an efficient hierarchical manner.

1.3.2. The Natural Language Component of ADAPT

Communication between humans and the robot will be assured by a natural language system implemented within the Soar cognitive modeling framework [26,27]. The natural language program in Soar is called NL-Soar (http://linguistics.byu.edu/nlsoar/homepage.html). Development of Soar's natural language capability, originally begun at Carnegie-Mellon University, is now centered at Brigham Young University under the direction of Deryle Lonsdale. This system has been previously integrated in other Soar-based task modeling situations involving such agent contexts as the NASA test director [43,44], intelligent forces in combat situations [25,51], simultaneous interpreters [28] and language learners [58].

The system supports spoken human language input via an interface with Sphinx, a popular speech recognition engine [16,24]. Textual inputs representing best-guess transcriptions from the system will be pipelined whole utterances into the main natural language component.

The NL comprehension component processes each word individually and performs the following operations via Soar operators in order to understand the input text:

- lexical access (which retrieves morphological, syntactic, and semantic information for each word from its lexicon) [30,31]
- syntactic model construction (linking together pieces of an X-bar parse tree) [51]
- semantic model construction (fusing together pieces of a lexical-conceptual structure) [52]
- discourse model construction (extracting global coherence from individual utterances) [18,19]

Each of these functions is performed either deliberately (by subgoaling and via the implementation of several types of lower-level operators) or by recognition (if chunks and pre-existing operators have already been acquired via the system's learning capability). Three types of structure resulting from the utterance will be important for subsequent processing: the X-bar model of syntax, the LCS model of semantics, and the discourse model. Work on this project will extend our prior work in resolving word sense, syntactic, and semantic ambiguities using symbolic [40], exemplar-based [19,55], and hybrid [32] architectures. The depth and breadth of the interface between these structures and the robot's incoming percepts and outgoing actions will be explored during the project. The NL generation component uses all of these components in reverse order to generate a textual utterance from planned semantic content [29].

The top-level language operators mentioned above can be sequenced in an agent's behavior with each other and with any other non-language task operators, providing a highly reactive, interruptible, interleavable real-time language capability [43]. In agent/human dialogues these properties are crucial [1,28].

As is typically implemented for human/robotic interaction, our system uses dialogue-based discourse interface between the robot and the NL component. The system's discourse processing involves aspects of input text comprehension (including referring to the prior results of syntax and semantics where necessary) and generation (i.e. the production of linguistic utterances). Both applications of discourse processing will involve planning and plan recognition, linguistic principles, real-world knowledge, and interaction management. The robotics domain will require a limited command vocabulary size of some 1500 words initially, and utterances will be comparatively straightforward. This will also improve the recognition rate of the speech engine and support more diverse interaction environments.

Possible communication modalities include [34]: (1) a command mode, or task specification, where the robot is capable of understanding imperative sentences and acting upon them (e.g. “Turn ninety-degrees to the left.” “Close the door.”) Other, more difficult types of utterances include (2) execution monitoring, where the robot takes instructions as part of a team, (3) explanation/error recovery to help the robot adapt and cooperate with changing plans, and (4) updating the environment representation to allow a user to aid the robot in maintaining a world model beyond its own perceptions. To begin with, the robot will understand imperative utterances, but other types of comprehension capabilities, as well as language generation, will be incrementally added.

Using dialogue processing in the human/robot interface allows, but also requires, the robot to maintain a model of the world and to maintain a record of the dialogue. Without a discourse/dialogue component, utterances would be difficult to connect to the robot’s environment. Appropriate modeling approaches include: finite-state automata (FSA's), frame/task-based dialogues, belief-desire-intention (BDI) architectures, and plan recognition constructs [48]. To varying extents, they integrate comprehension and generation capabilities to assure coherent interaction.

FSA models, while fairly straightforward to implement, are completely deterministic and do not require the robot to manage an extensive worldview or model of dialogues. They also severely restrict the robot's ability to adapt to a wide variety of communicative situations. Frame-based systems relax strict determinism [56], but still are not as flexible as highly dynamic environments may require. BDI architectures [23] allow a robot to model the discourse participants' beliefs and goals, as well as its perceptions of the same. A dialogue move engine (DME) selects new moves depending on the context of utterances, the goals, and the evolving conversational model. As a standalone component, though, BDI/DME processing does not allow a tight enough integration with the Soar environment to permit learning, and operator-based processing.

NL-Soar implements a discourse recipe-based model (DRM) for dialogue comprehension and generation [18]. It learns the discourse recipes, which are generalizations of an agent’s discourse plans, as a side effect of dialogue planning. This way, plans can be used for comprehension and generation. If no recipe can be matched, the system resorts to dialogue plans. This allows both a top-down and bottom-up approach to dialogue modeling. It also supports elements of BDI/DME functionality such as maintaining a common ground with information about shared background knowledge and a conversational record.

Initiative is an important aspect in dialogue. Different approaches to managing dialogue vary from system-initiative (where the robot controls interaction) to user-initiative (where the human controls interaction) to, ideally, mixed or joint-initiative (where the robot and the human take turns controlling and relinquishing control as situations unfold). FSA and frame-based implementations support system-initiative applications, but a highly reactive robot requires mixed initiative. Part of the work in this project will involve investigating and demonstrating the relative advantages and

disadvantages of BDI vs. DRM approaches for supporting (successively) human-, system-, and mixed-initiative robotic interactions.

1.4. Task Definition for Evaluating ADAPT

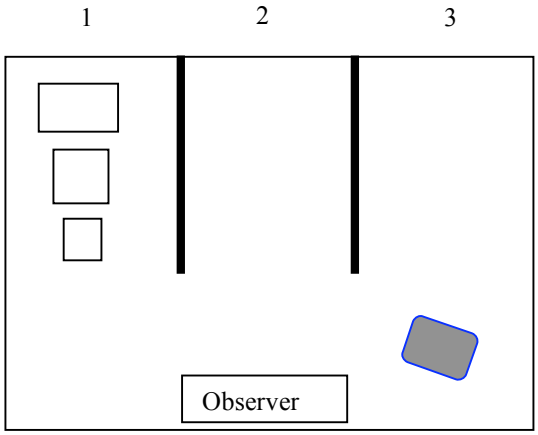
The goal of the ADAPT architecture is to increase the robot's comprehension of its environment. Thus, our project is not focused on a single task such as finding land mines. Instead, we have selected a flexible class of mobile robot applications as our example domain: the "shepherding" class. In this application, one or more mobile robots have the objective of moving one or more objects so that they are grouped according to a given constraint. An example from this domain is for a single mobile platform to push an object from its start position over intermediate terrain of undefined characteristics into an enclosure. Another example is for it to push a set of objects of various shapes and size all into the enclosure. A more complex example is to group objects so that only objects of a particular color are in the enclosure.

This class of tasks is attractive for two reasons. The first is that it includes the full range of problems for the robot to solve, from abstract task planning to real-time scheduling of motions, and including perception, navigation, communication with humans and grasping of objects. In addition, the robot must learn how to push one or more objects properly. This range of demands is ideal for our purposes, because it creates a situation in which complex hierarchies of features and constraints arise. Also, the tasks can be made dynamic by including objects that can move by themselves, e.g. balls or other robots.

The second reason is that we can embed lots of other problems in it. For example, we can embed bin-packing problems by making them enclosure-packing problems. Also, we can embed sorting problems and block-stacking problems. This creates a situation in which the robot can be presented with well-known computing tasks in a real setting with perceptual and navigational difficulties, rather than just as abstract tasks.

2. A SIMPLE EXAMPLE

This is an example of a shepherding task and how reformulation can make it easier to solve.



The robot (gray) must move the three boxes from the leftmost column to the rightmost. It can push or lift one box at a time. A box can never be in front of a smaller box, as seen from the observer. The larger a box is, the taller also. The front area must be kept clear so the robot can move; there can be no boxes left in this area.

An abstraction hierarchy for this task will contain a number of levels ranging from low-level features and actions such as "find the right front edge of the small box" to abstract features and actions such as "move the small and middle boxes to the middle enclosure". There are many possible such hierarchies for this task. For example, consider the abstract action of moving one box from one enclosure to another. The robot could index the possible instances of this action by the box that is being moved. This in effect creates new abstractions, e.g. "moving the small box one enclosure over", which would be seen by the robot as the same whether it is moving the box from enclosure 1 to enclosure 2, or from enclosure 2 to enclosure 3. This is a useful abstraction.

But the robot could also choose to index these actions by the enclosures, creating different abstractions, e.g. moving any box from enclosure 1 to enclosure 2, regardless of which box it is. This is a less useful abstraction, and not just because of the details of gripping different size boxes. As we show in [13,14], this second method of indexing actions does not scale up, so that once the robot learns the solution for three boxes, it will not be able to reuse what it has learned to

move four boxes correctly. However, the previous method of indexing the actions (by the box moved) does scale up to all larger problems, so it is a superior formulation of the task.

An even better abstract move is "move the small and middle boxes from one enclosure to another". This way of formulating the task is self-similar, by viewing the small and middle boxes together as one "logical box", with an associated invariant that is their relative position. Because of this self-similarity, this abstraction scales up to larger versions of the task. Furthermore, using this abstract move in planning eliminates subgoal interference in this task [10,12,13,14], making it a superior abstraction for this task.

The whole point is that the human designers should not be providing these abstractions to the robot. A truly autonomous robot must be able to generate and choose its own problem-solving abstractions. ADAPT does this by reformulating its schema hierarchy. The search in the space of formulation is far more expensive than just searching the moves of the three boxes, but the formulation search has a far higher utility as its solution solves a whole class of tasks rather than just one.

The upper layers of the abstraction hierarchy for the above example task are isomorphic to the three-disk Towers of Hanoi puzzle. This is a simple illustration of how we can embed known tasks into shepherding tasks. Many applications for mobile robots can be cast as shepherding tasks, including agricultural robotics applications, routine material transport in factories, warehouses, office buildings and hospitals, indoor and outdoor security patrols, inventory verification, hazardous material handling, hazardous site cleanup, inspection of hazardous waste storage sites, and numerous military applications.

3. SUMMARY

We are designing and implementing ADAPT, which is a unified cognitive architecture for a mobile robot. The goal of this project is to endow a robot with the full range of cognitive abilities, including perception, use of natural language, learning and the ability to solve complex problems. The perspective of this work is that an architecture based on a unified theory of robot cognition has the best chance of attaining human-level performance.

ADAPT is based on an integration of three theories: a theory of cognition embodied in the Soar system, the RS formal model of sensorimotor activity and an algebraic theory of decomposition and reformulation. These three theories share a hierarchical structure that is the basis of their integration. Soar models cognition as subgoaling in a hierarchy of problem spaces; the RS model consists of a hierarchy of sensorimotor modules. The theory of reformulation provides the method of synthesis of such hierarchies by finding one or more local patterns that replicate at different scales, creating a representational hierarchy that is simply expressed yet capable of great complexity. This structure underlies all of the robot's principal cognitive activities: its perception, its use of natural language, and its planning and problem solving.

ADAPT's world model is a powerful 3D multimedia engine capable of rendering complex, dynamic environments. The world model functions as the interpretation of the symbols in Soar's working memory, and gives the robot the ability to visualize alternatives.

REFERENCES

1. G. Aist, J. Dowding, B. A. Hockey, M. Rayner, J. Hieronymus, D. Bohus, B. Boven, N. Blaylock, E. Campana, S. Early, G. Gorrell, and S. Phan. "Talking through procedures: An intelligent Space Station procedure assistant," In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03), Budapest, Hungary, 2003.
2. Arkin, R., *Behavior-Based Robotics*, MIT Press, Cambridge, MA, 1998.
3. Michael F. Barnsley, "Fractal Image Compression", Notices of the American Mathematical Society, Volume 43, Number 6, pp. 657-662, June, 1996.
4. Benjamin, D. Paul, Damian Lyons and Deryle Lonsdale, "Designing a Robot Cognitive Architecture with Concurrency and Active Perception", Proceedings of the AAAI Fall Symposium on the Intersection of Cognitive Science and Robotics, Washington, D.C., October, 2004.
5. Benjamin, D. Paul, Damian Lyons and Deryle Lonsdale, "ADAPT: A Cognitive Architecture for Robotics", Proceedings of the International Conference on Cognitive Modeling, (ICCM-2004) Pittsburgh, Pa., July, 2004.
6. Benjamin, D. Paul, Damian Lyons and Deryle Lonsdale, " Cognitive Robots: Integrating Perception, Action and Problem Solving in Behavior-Based Robots", AAMAS-2004 Proceedings, pp. 1308-1309, New York City, July, 2004.

7. Benjamin, D. Paul, “On the Emergence of Intelligent Global Behaviors from Simple Local Actions”, Journal of Systems Science, special issue: Emergent Properties of Complex Systems, Vol. 31, No. 7, 2000, 861-872.
8. Benjamin, D. Paul, “Decomposing Robotic Representations by Identifying Local Symmetries and Invariants”, Proceedings of the 1998 NSF Design and Manufacturing Grantees Conference, Monterrey, Mexico, 1998.
9. Benjamin, D. Paul, “A Decomposition Approach to Solving Distributed Constraint Satisfaction Problems”, Proceedings of the IEEE Seventh Annual Dual-use Technologies & Applications Conference, IEEE Computer Society Press, 1997.
10. Benjamin, D. Paul, “Transforming System Formulations in Robotics for Efficient Perception and Planning”, Proceedings of the IEEE International Symposia on Intelligence and Systems, Washington, D.C., IEEE Computer Society Press, 1996.
11. Benjamin, D. Paul, “Behavior-preserving Transformations of System Formulations”, Proceedings of the AAAI Spring Symposium on Learning Dynamical Systems, Stanford University, March, 1996.
12. Benjamin, D. Paul, “Formulating Domain Theories for Software Design”, Proceedings of the IEEE Sixth Annual Dual-use Technologies & Applications Conference, IEEE Computer Society Press, 1996.
13. Benjamin, D. Paul, “Formulating Patterns in Problem Solving”, *Annals of Mathematics and AI*, **10**, pp.1-23, 1994.
14. Benjamin, D. Paul, “Reformulating Path Planning Problems by Task-preserving Abstraction”, *Journal of Robotics and Autonomous Systems*, **9**, pp.1-9, 1992.
15. A. Blake and A. Yuille, eds, *Active Vision* , MIT Press, Cambridge, MA, 1992.
28. Nate Blaylock, James Allen, and George Ferguson, “Synchronization in an asynchronous agent-based architecture for dialogue systems,” In Proceedings of the 3rd SIGdial Workshop on Discourse and Dialog, Philadelphia, 2002.
16. L. Chase, R. Rosenfeld, A. Hauptmann, M. Ravishankar, E. Thayer, P. Placeway, R. Weide, and C. Lu. “Improvements in language, lexical, and phonetic modeling in Sphinx-II,” In Proc. Spoken Language Systems Technology Workshop. Morgan Kaufmann Publishers, 1995.
17. Eilenberg, Samuel, *Automata, Languages, and Machines,* Volumes A and B, Academic Press, 1976.
18. Green, N. and Lehman, J.F., “An Integrated Discourse Recipe-Based Model for Task-Oriented Dialogue.” *Discourse Processes, 33*(2), 2002.
19. Daniel Jones, "Analogical Natural Language Processing", Studies in Computational Linguistics, UCL Press, London and Bristol, Pa., 1996.
20. Laird, John E., Rosenbloom, Paul S., and Newell, Allen, “Towards Chunking as a General Learning Mechanism,” AAAI-84, 1984.
21. Laird, J.E., Newell, A. and Rosenbloom, P.S., “Soar: An Architecture for General Intelligence”, *Artificial Intelligence* **33**, pp.1-64, 1987.
22. Lallement, Gerard, *Semigroups and Combinatorial Applications*, Wiley & Sons, 1979.
23. Larsson, S., Ljunglöf, P., Cooper, R., Engdahl, E., and Ericsson, S., “GoDiS—an Accommodating Dialogue System”, In *Proceedings of ANLP/NAACL-2000 Workshop on Conversational Systems*, Seattle, 2000.
24. K. F. Lee, “Automatic Speech Recognition: The Development of the SPHINX SYSTEM,” Kluwer Academic Publishers, Boston, 1989.
25. Lehman, J., VanDyke, J., and Rubinoff, R., “Natural language processing for IFORs: comprehension and generation in the air combat domain”, In *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, 1995.
26. Lehman, J., VanDyke, J. Lonsdale, D., Green, N., and Smith, M., “A Building Block Approach to a Unified Language Capability,” Unpublished manuscript, 1997.
27. Lehman, J.F., Laird, J.E., and Rosenbloom, P.S., “A Gentle Introduction to Soar, an Architecture for Human Cognition,” Invitation to Cognitive Science, vol. 4, Sternberg & Scarborough (eds.), MIT Press. 1995.
28. Lonsdale, Deryle, “Modeling Cognition in SI: Methodological Issues,” *International journal of research and practice in interpreting*, Vol. 2, no. 1/2, pages 91-117; John Benjamins Publishing Company, Amsterdam, Netherlands, 1997.
29. Lonsdale, Deryle, “Leveraging Analysis Operators in Incremental Generation,” *Analysis for Generation: Proceedings of a Workshop at the First International Natural Language Generation Conference*, pp. 9-13; Association for Computational Linguistics; New Brunswick, NJ, 2000.
30. Lonsdale, Deryle, “An Operator-based Integration of Comprehension and Production,” *LACUS Forum XXVII*, pages 123–132, Linguistic Association of Canada and the United States, 2001.

31. Deryle Lonsdale and C. Anton Rytting, “Integrating WordNet with NL-Soar,” *WordNet and other lexical resources: Applications, extensions, and customizations*; Proceedings of NAACL-2001; Association for Computational Linguistics, 2001.
32. Deryle Lonsdale and Michael Manookin, "Combining Learning Approaches for Incremental On-line Parsing", Proceedings of the Sixth International Conference on Conceptual Modeling, pp. 160-165, Lawrence Erlbaum Associates, 2004.
33. Deryle Lonsdale, “Modeling cognition in SI: Methodological issues. International Journal of Research and Practice in Interpreting”, **2(1/2)**: 91–117, 1997.
34. Lueth, T.C., Laengle, T., Herzog, G., Stopp, E., and Rembold, U., “KANTRA: Human-Machine Interaction for Intelligent Robots Using Natural Language,” In *Proceedings of the 3rd International Workshop on Robot and Human Communication.* Nagoya, Japan, 1994.
35. Lyons, D.M. and Hendriks, A., “Planning as Incremental Adaptation of a Reactive System”, Journal of Robotics & Autonomous Systems **14**, 1995, pp.255-288.
36. Lyons, D.M. and Hendriks, A., “Exploiting Patterns of Interaction to Select Reactions”, Special Issue on Computational Theories of Interaction, Artificial Intelligence **73**, 1995, pp.117-148.
37. Lyons, D.M., “Representing and Analysing Action Plans as Networks of Concurrent Processes”, IEEE Transactions on Robotics and Automation, June 1993.
38. Lyons, D.M. and Arbib, M.A., “A Formal Model of Computation for Sensory-based Robotics”, IEEE Transactions on Robotics and Automation **5**(3), Jun. 1989.
39. Lyons, D., and Arkin, R.C., “Towards Performance Guarantees for Emergent Behavior”, (Submitted) IEEE Int. Conf. on Robotics and Automation, New Orleans LA, April 2004.
40. Michael Manookin and Deryle Lonsdale, "Resolving Automatic Prepositional Phrase Attachments by Non-Statistical Means", LACUS Forum XXX: Language, Thought, and Reality, The Linguistic Association of Canada and the United States, pp. 301-312, 2004.
41. Stacy Marsella, Jonathan Gratch and Jeff Rickel, "Expressive Behaviors for Virtual Worlds," Life-like Characters Tools, Affective Functions and Applications, Helmut Prendinger and Mitsuru Ishizuka (Editors), Springer Cognitive Technologies Series, 2003.
42. Miller, C. S. (1993), “Modeling Concept Acquisition in the Context of a Unified Theory of Cognition”, EECS, Ann Arbor, University of Michigan.
43. Nelson, G., Lehman, J.F., and John, B.E., “Experiences in interruptible language processing,” Proceedings of the AAAI Spring Symposium on Active Natural Language Processing, Stanford, CA,21-23 March,1994.
44. Nelson, G., Lehman, J.F., and John, B.E., “Integrating cognitive capabilities in a real-time task,” In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society.* Atlanta, GA, August 1994.
45. Newell, Allen, *Unified Theories of Cognition*, Harvard University Press, Cambridge, Massachusetts, 1990.
46. M. Nicolescu and M. Mataric, “Extending Behavior-based System Capabilities Using an Abstract Behavior Representation”, *Working Notes of the AAAI Fall Symposium on Parallel Cognition*, pages 27-34, North Falmouth, MA, November 3-5, 2000.
47. Petrich, Mario, *Inverse Semigroups*, John Wiley & Sons, Inc., New York, 1984.
48. Rees, Rebecca, “Investigating Dialogue Managers: Building and Comparing FSA Models to BDI Architectures, and the Advantages to Modeling Human Cognition in Dialogue,” Honors Thesis, BYU Physics Department, 2002.
49. P. S. Rosenbloom, J. E. Laird, and A. Newell, *The SOAR Papers* . MIT Press, 1993.
50. Rosenbloom, P.S., Johnson, W.L., Jones, R.M., Koss, F., Laird, J.E., Lehman, J.F., Rubinoff, R., Schwamb, K.B., and Tambe, M., “Intelligent Automated Agents for Tactical Air Simulation: A Progress Report”, Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation, pp.69-78, 1994.
51. Rubinoff, R. and Lehman, J. F., “Natural Language Processing in an IFOR Pilot,” Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL., 1994.
52. C. Anton Rytting and Deryle Lonsdale, “An Operator-based Account of Semantic Processing,” *The Acquisition and Representation of Word Meaning*; pp. 84-92; European Summer School for Logic, Language and Information, Helsinki, 2001.
53. S. Schneider, *Concurrent and Real-time Systems: The CSP Approach*. Wiley, 1999.
54. M. Senechal, *Quasicrystals and Geometry*, Cambridge University Press, 1995.
55. Royal Skousen, Deryle Lonsdale, and Dilworth B. Parkinson, "Analogical Modeling: An Exemplar-Based Approach to Language", Human Cognitive Processing Series, Vol. 10, John Benjamins Publishing Co., Amsterdam, 2002.
56. Spiliotopoulos, D., Androutsopoulos, I., and Spyropoulos, C. D., “Human-Robot Interaction Based on Spoken

Natural Language Dialogue," In *Proceedings of the European Workshop on Service and Humanoid Robots (ServiceRob '2001)*, Santorini, Greece, 25-27 June 2001.
57. Martha Steenstrup, Michael A. Arbib, Ernest G. Manes, *Port Automata and the Algebra of Concurrent Processes*. JCSS 27(1): 29-50, 1983.
58. Van Dyke, J. and Lehman, J.F., "An architectural account of errors in foreign language learning," In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society,* 1997.
59. Wray, R. E., and Chong, R. S., Explorations of quantitative category learning with Symbolic Concept Acquisition. 5th International Conference on Cognitive Modeling (ICCM), Bamberg, Germany, 2003.