1. Introduction: Cognition and Robotics

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 [73]. Complex tasks that involve reasoning about spatial and temporal relationships require robots to possess more advanced mechanisms for planning, reasoning, learning and representation. Tasks that involve interaction with humans, especially non-expert humans, require a sophisticated understanding of natural language.

A truly cognitive architecture has not yet been implemented in robotics. Robots have been programmed to perform specific tasks such as mowing the lawn or navigating in the desert, but they cannot act autonomously to choose tasks and devise ways to perform them. Even when performing their allotted tasks, they lack flexibility in reacting to unforeseen situations. Currently, the design of all the important perceptual and decision-making structures is done by the programmers before the robot begins its task. The semantics for the symbols and structures the robot uses are determined and fixed by these programmers. This leads to fragmented abilities and brittle performance. The robots cannot adapt their knowledge to the task, cannot solve tasks that are even slightly different from those they have been programmed to solve, cannot communicate effectively with humans about their goals and performance, and just don't seem to understand their environment. This is the principal stumbling block that prevents robots from achieving high levels of performance on complex tasks, especially tasks involving interaction with people.

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 that is building 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 project is an interdisciplinary, multi-university collaboration involving three professors and over a dozen graduate and undergraduate students, many of whom are members of groups underrepresented in scientific research.

We have implemented a robot control language called RS (Robot Schemas) within a mature cognitive model, the Soar cognitive architecture. This serves as the basis of the ADAPT architecture. Our goal is to create a new kind of robot architecture capable of robust behavior in unstructured environments, exhibiting the full range of cognitive abilities: problem solving and planning skills, learning from experience, novel methods of perception, comprehension of natural language and speech generation. Thus, this project has great potential to improve the capabilities of robots in many areas, e.g. robots that aid the handicapped as well as robots in the military.

RS is a powerful language for distributed, concurrent, real-time robot programming that has previously been implemented and tested successfully in factory automation. Soar is a widely used cognitive architecture based on a large body of cognitive science research that has demonstrated success over the past twenty years in problem solving and learning on a wide range of tasks.

The basis for the integration of these systems is the linguistic structure underlying both perception and motion. Algebraic linguistics provides a theory of the local neighborhood structure of languages. This theory is applicable to languages of actions (motion) and languages of perceptual transformations. In this way, this theory provides algorithms for the transformation and reformulation of percepts and motions. In particular, this theory provides a method of automatically synthesizing a hierarchy of sensory-motor schemas for a given situation. This theory forms the basis for the robot's planner and also provides the mapping of the semantics of RS into Soar.

1.1. The Goals of the ADAPT Project

The central research goal in the development of this architecture is the examination of the interrelationship between perception, problem solving and the use of natural language in complex, dynamic environments. In most work in cognitive modeling, these three aspects of cognition are treated

separately. In particular, perception is usually a peripheral activity that is clearly separated from problem solving. And most previous work in problem solving and robotics has constructed systems by combining separate modules for each capability, without a principled basis for their integration.

We want to explore issues such as how perception is related to representation change in problem solving, and how linguistic structures may affect perception and problem solving. ADAPT is an architecture intended to explore the integration of perception, problem solving and natural language at a deeper structural level.

We believe that the integration of these capabilities must stem from a central organizing principle, and in ADAPT that principle is language. Language provides not only the means of interaction between people and ADAPT, but also provides the basis for the integration of perception and problem solving. ADAPT's method of problem solving is to reformulate problems to exploit their self-similar structure. It views the robot's actions and the actions occurring in the environment as a language, and uses methods from algebraic linguistics to analyze local symmetries and invariants of that language. These local invariants are then used to synthesize simple programs (schemas) that act at multiple scales to solve complex problems [11, 22, 23].

ADAPT's view of perception is that it is the process of creating and modifying linguistic formulations. Perception is thus a problem to be solved by the cognitive architecture just as it solves other problems.

Our approach to the design of these formulations is to analyze the dynamic structure underlying perception and action for a given task [11]. The mathematical tools for this come from the algebraic linguistics and the theory of semigroups. We use these tools to extract from this structure a hierarchy of local neighborhoods for the task. Each neighborhood possesses a local coordinate system that specifies perception and motion within the neighborhood. Under certain common conditions, these neighborhoods form a self-similar fractal structure known as a substitution tiling. When this occurs, the structure of the neighborhoods is replicated at many scales. This structure can be used to synthesize a hierarchy of sensory-motor schemas for many classes of tasks.

We use this formal approach to integrate RS (Robot Schemas) [64], which is a programming language for real-time distributed robot control, and Soar [41, 72], which is a mature cognitive architecture. Both possess a detailed model of hierarchical control based on local neighborhoods. In Soar, these neighborhoods are problem spaces, and the mechanism of universal subgoaling exploits the recursive structure of a substitution tiling. RS possesses a formal algebraic semantics based on hierarchies of port automata, which correspond exactly to the neighborhood structure. We have implemented RS automata as Soar problem spaces; this is the core of ADAPT's cognitive mechanism.

ADAPT is under development on Pioneer robots in the Pace University Robotics Lab and the Fordham University Robotics Lab. Publications describing our work on ADAPT are [7, 8, 9, 10].

1.2. Roadmap of this Proposal

In the next subsection, we give an overview of previous related work and a brief description our previous NSF-sponsored research. This is followed by the four main sections of the proposal.

In section 2, we present the RS/Soar core of ADAPT, describing its implementation and the rationale behind its design. In section 3, we discuss the world model. ADAPT has a distinctive world model and uses the model in a number of novel ways to provide ADAPT with sophisticated abilities. We describe some of these, including the predictive vision system and use of image schemas to provide semantics for language. In section 4, we describe the class of tasks we are using to test our system, and section 5 describes our research and coordination plan.

1.3. Previous Work and NSF-Sponsored Research

Early work in robotics reflected the first crucial robotic application – manufacturing. Thus, much of the work involved the detailed analysis of geometric (CAD-like) descriptions and the selection of an action sequence tailored for that geometry [28]. If any sensing was involved, it was all carried out first, followed by extensive geometric analysis, followed at last by action. Although this approach functioned quite well in the controlled and restricted environments seen in manufacturing, it did not generalize well

to arbitrary, unstructured environments. It was too slow and prone to failure when the environment could not be exactly modeled or known. Behavior-based robotics [29] was first proposed by Brooks in the mid 1980's as an alternative approach that emphasized fast, reactive actions and the absence of explicit models. Behavior-based systems cope well with unstructured environments. However, the performance level of a behavior-based system is heavily dependent on the skill of the behavior "programmer" who builds it. Furthermore, behavior-based systems are not easily integrated with deliberative reasoning. Hybrid systems were developed therefore to address the issue of integrating reactive and deliberative action systems [66].

In [61,62] Lyons and Hendriks present an approach to integrating robot planning and reaction. RS is used to specify and implement the reactive component of the robot system and ITL [2] is used for the planning component. Action schemas in ITL are linked to pre-packaged RS automata networks. The planner reasons about instantiating and connecting networks to achieve its goals. As a plan is created, the planner incrementally transmits the instructions to the reactive component, which instantiates automata networks and begins the execution of the plan concurrently with the ongoing elaboration of the plan. These networks include monitoring automata that report to the planner when components of the plan finish execution, or when sensed conditions change to such an extent that the plan is no longer valid. The system could respond to the violation of assumptions about the environment by replanning, again concurrent with the ongoing automata execution, and upgrade components of the plan in a safe fashion. The system was limited however to the set of action schemas and automata provided to the planner.

This limitation is typical of planning systems and has led to research on systems that can autonomously expand or adapt their repertoire of planning components. Two such approaches are to add a learning capability so that the system can learn new plan components, or to add an ability to change formulations so that the system can transform a task into a much more easily solved form. Much research has added a learning module or a change of representation module to an existing planner in an attempt to construct a planner of greater power and generality. Typical examples of this line of research are [3,32,33,38,53,85].

Our approach is fundamentally different. Existing hybrid architectures consist of separate components that have been connected, e.g. a planner, a reactive component, a natural language component, a vision component. These components usually derive from different assumptions and methodologies, and the connections between components are merely that the results of each component are available to the other components. There is no deeper integration, e.g. how the architecture sees bears no relation to how it solves problems. We believe that we are much more likely to achieve sophisticated robot performance in unstructured environments if we build a unified architecture in which problem solving, perception, real-time action, use of natural language and learning are based on a single representational structure and a single approach to problem solving. So our approach is to build a unified cognitive robot architecture.

In a previous NSF SGER grant [26], Benjamin constructed the basics of an algebraic theory of task decomposition and reformulation and explored its application to navigation of a small mobile robot. That project consisted mainly of the development of the algebra, together with an initial demonstration of its application to robotics. The robot used was a Pioneer P1, and the decomposition theory was implemented in Saphira, which is the C++-based control software supplied with that robot. The basics of the algebra were completely worked out and the P1 robot was successfully navigated using handcoded patterns derived from the theory. This work was documented in [11-26], and forms the basis of the reformulation capability in this project.

This work has been continued under a Department of Energy grant [25], using a more advanced robot, the Pioneer P2. Soar was selected to control the robot, because of its subgoaling and learning capability, and also because of its potential to model a significant portion of human cognition. Soar was connected to Saphira, and successful subgoaling and autonomous learning of navigation has been demonstrated using the method of analyzing local symmetries and invariants, which is described later in this proposal.

In other NSF-sponsored research, one of us has participated in a formal methods curriculum development project at Pace University [24]. This project involved teaching a sequence of three courses to undergraduate students in computer science. In these courses, topics in data structures and discrete

mathematics were taught using the Z (zed) tools. Students were taught how to use these formal tools to analyze their code and provide feedback for design. We taught these courses for a few years and compared the performance of students in these sections with those in traditional sections of the same courses (taught without formal methods tools). The students taught formal methods showed a clear performance improvement, and incidentally had a higher rate of retention.

2. Robot Schemas Embedded in a Cognitively Plausible Substrate

The core of ADAPT consists of a formal integration of RS and Soar. RS serves as the higher-level language of ADAPT and Soar as its implementation substrate, so the first two subsections describe RS and then Soar. The third subsection discusses how RS is implemented in Soar, and the final two subsections give an overview of NL-Soar and the method of synthesizing schema hierarchies.

2.1. RS

RS is a formal model of robot computation developed by Damian Lyons that is based on the semantics of networks of port automata. This model is realized in the RS programming language, which is used to specify a network of processes. A port automaton (PA) is a finite-state automaton equipped with a set of synchronous communication ports [64]. 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, which represent the semantics of processes. This vocabulary is based on the idea of a *schema*. A schema is a pattern of perception and action [5]. Each schema has a sensory part and a motor part, representing the goal of executing a particular motion depending on certain perceptual conditions. Schemas also contain explicit qualitative temporal information, e.g. about intervals of time during which an action must take place. Schemas are instantiated to create schema instances that are connected to each other to form a network. 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 [80]. 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 [63,64]. 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 [65].

An RS program creates and manages a hierarchy of schemas. The great strength of RS is the formal semantics it provides for composition of schemas, which guarantees the soundness of the hierarchy.

RS lacks a synthesis mechanism and a learning mechanism, as well as a model for implementation of abilities such as learning and language. And although schemas have a general psychological plausibility, there is no real cognitive plausibility for the RS language. For these reasons, we have implemented RS in a powerful and general cognitive architecture.

2.2. Soar

Soar [41, 72] 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. Further problem spaces govern other aspects of the system, including use of natural language, sonar interpretation, and VSLAM.

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 [38].

The Soar publications extensively document how this learning method is cognitively plausible, because it speeds up the system's response time in a manner that accurately models the speedup of human subjects on the same tasks [40]. Also, the Soar research community is implementing a wide range of aspects of cognition in Soar, including natural language [57, 59], concept learning [69, 86] and emotion [68]. But Soar lacks a comprehensive theory of parallelism, which hampers its usefulness in robotics.

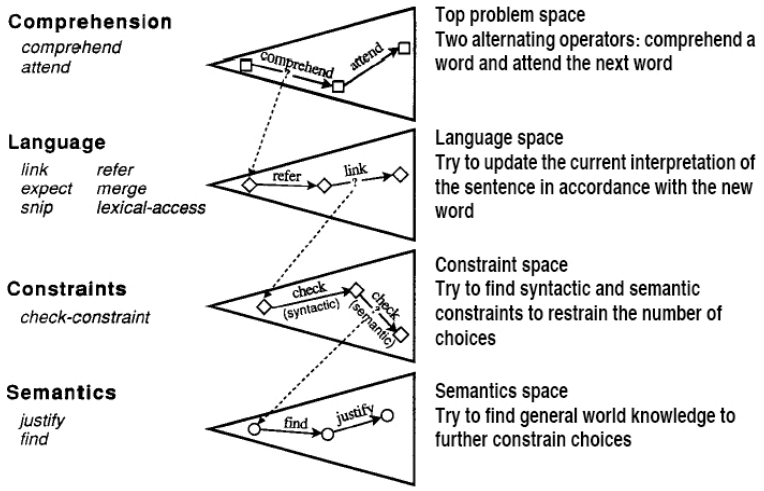

Figure 1. An example goal stack from NL-Soar (from Lehman, Lewis, Newell and Pelton, 1991).

The natural language program in Soar is called NL-Soar. 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 [70,71], intelligent forces in combat situations [48,76,77], simultaneous interpreters [54] and language learners [84].

Figure 1 shows some of the problem spaces from NL-Soar. In this figure, Soar subgoals from the Comprehension problem space to the Language problem space, and then to the Constraints and Semantics problem spaces. These spaces are not arranged in a fixed hierarchy; Soar can subgoal from any space to any others in any order to search for an accurate understanding of an utterance.

2.3. The Implementation of RS in Soar

Soar manipulates a hierarchy of problem spaces, and the formal semantics of RS consists of a hierarchy of port automata [83]. 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. RS becomes the higher-level language of ADAPT, and Soar is its assembly language. In this way, ADAPT inherits the abilities of both RS and Soar: ADAPT subgoals and chunks exactly as Soar does, while executing concurrent, real-time RS programs.

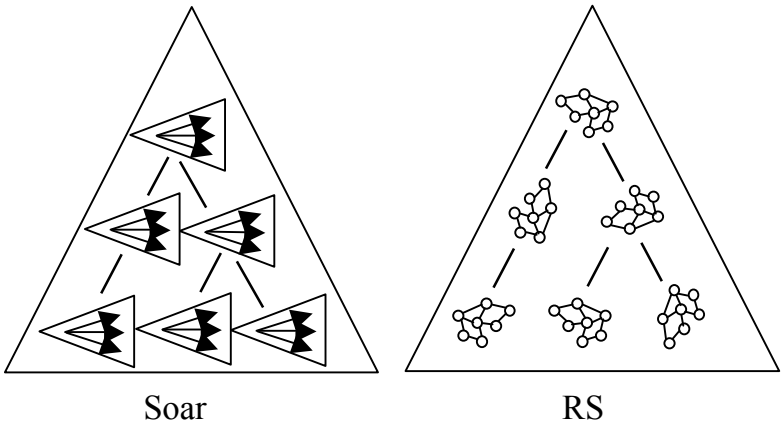

Figure 2. Soar problem spaces implement RS port automata.

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, and its combination of deliberative planning and reactive behavior. Soar provides a cognitive model with a full range of cognitive abilities, including perception, deliberative planning, reaction, natural language, learning, and emotion.

By integrating RS and Soar, we have created an architecture that:

- processes a wide range of modes of perceptual input,
- provides a complete range of cognitive abilities, including language and learning,
- reasons about time and resources,
- implements parallel, reactive behaviors, and
- learns new high-level sensors and behaviors.

ADAPT contains Soar problem spaces that can reason about the automata networks that are used to represent the environment and the task, and generate abstract plans that are refined into automata networks, creating new networks as necessary. The basic sensory schemas are always on and are shallow and bottom-up. Because ADAPT can instantiate new automata networks when necessary, it can add networks to the sensory system that are task-specific, deep, and top-down.

ADAPT's schema hierarchy is attached to the top state in Soar. ADAPT uses Soar's subgoaling to create its hierarchy of schemas, via operators that select, instantiate and interconnect schemas. This permits Soar's chunking method to learn how to build, monitor and modify networks of schemas.

General schemas are kept permanently in working memory, so that they function as a kind of declarative memory. Each schema is linked to other schemas that are relevant. This library of schemas in working memory that can be instantiated and activated is similar to the declarative memory in which ACT-R [4] stores its chunks. We follow the example of ACT-R and permit ADAPT to create links between schemas to reflect relationships between them, such as “is a generalization of” and “is the opposite of”, as well as activation links labeled by weights that can be used by spreading activation.

Schemas have *input ports* and *output ports*. The ports of schema instances are connected to create ADAPT's network. Schemas can be decomposed into an assemblage of schemas that represent the world in more detail. Such *assemblage schemas* are attached to their parent schemas, creating a hierarchical structure that represents the world at varying granularities. Schemas that can be directly carried out by our Pioneer robots’ Aria software possess an implementation attribute that contains the robot commands.

Each schema instance in the hierarchy has a weight and a priority, which are used by ADAPT to compute what the robot will actually do. ADAPT does not select a single schema to execute on the robot. Instead, all active schemas that possess implementations are *blended* by ADAPT's *resolver* to create the actions actually executed on the robot platform. Blending involves weighting each schema

implementation by its weight, adding together weighted implementations of equal priority, then disabling lower-priority actions if they contradict higher-priority actions.

For example, suppose a "move-forward" schema and a "turn-right" schema both have equal priority, with "move-forward" weighted by 0.9 and "turn-right" weighted by 0.1". The implementation of "move-forward" is for both wheels to move at speed 100, and the implementation of "turn-right" is for the right wheel not to move and the left wheel to move at speed 100. Then the blend of these two schemas will move the left wheel at speed 100 and the right wheel at speed 90, resulting in a slight angling to the right. If another move schema has a lower priority, it will be disabled. If another schema with a lower priority does not contradict the moves, e.g. a schema that tilts the camera, then it will be executed concurrently.

ADAPT subgoals in the same manner as Soar, but subgoaling does not necessarily choose one schema, but instead assigns weights for blending. Choosing one schema to execute is simply making one schema's weight 1 and all others' weights 0.

ADAPT's chunking mechanism is that of Soar. It creates a new rule (chunk) that summarizes preconditions for a subgoal's result. If the subgoal is selecting, instantiating or removing a schema, then the chunk will perform the operation in a single step. If the subgoal operates on multiple schemas, then the chunk will perform all the schema operations at once; in this way ADAPT can learn truly concurrent operations. For example, a subgoal may be to keep an object in focus while turning right. ADAPT can instantiate a schema to turn the cameras left and a schema to turn right. The decisions to instantiate and start both schemas will appear in one chunk, thus encoding distributed, concurrent control.

2.4. NL-Soar: ADAPT's Natural Language System

Communication between humans and the robot is handled by NL-Soar, which is a natural language system implemented within the Soar cognitive modeling framework [49,50]. The system supports spoken human language input via an interface with Sphinx, a popular speech recognition engine [30,47]. Textual inputs representing best-guess transcriptions from the system are pipelined as whole utterances into the main natural language component.

NL-Soar 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) [56,57]
- syntactic model construction (linking together pieces of an X-bar parse tree) [52]
- semantic model construction (fusing together pieces of a lexical-conceptual structure) [78, 79]
- discourse model construction (extracting global coherence from individual utterances) [34,35]

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. Future work on this project will extend our prior work in resolving word sense, syntactic, and semantic ambiguities using symbolic [67], exemplar-based [39,81], and hybrid [58] 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. NL-Soar uses all of these components in reverse order to generate a textual utterance from planned semantic content [55].

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 [70], which in agent/human dialogues is crucial [1,27].

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 involve planning and plan recognition, linguistic principles, real-world knowledge, and

interaction management. The robotics domain requires a limited command vocabulary size of some 1500 words initially, and utterances are comparatively straightforward. This will also improve the recognition rate of the speech engine and support more diverse interaction environments.

Possible communication modalities include [60]: (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. The nature of this model is discussed in section 3 of this proposal. 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 [75]. 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 [82], but still are not as flexible as highly dynamic environments may require. BDI architectures [46] 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 [34]. 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.

2.5. Synthesis of Schema Hierarchies by Decomposition

Hierarchies are a powerful mechanism for the organization of system components, because they can reduce complexity exponentially. Newell [72] makes this argument as the basis for the creation of the Soar architecture. However, using hierarchies leads to two requirements:

1) The ability to accurately compose components to create larger components, and
2) An efficient mechanism for synthesizing a set of components that implement a given component.

The first requirement states the need for a clear semantics for composition. RS provides this [64]. The second requirement states the need for a method of schema decomposition. ADAPT's decomposition method is based on the linguistic structure underlying both perception and motion. The motions and

perceptual transformations of the robot form a language. Algebraic linguistics and the theory of semigroups provide a theory of the local neighborhood structure of languages [31, 42, 74]. We have shown how this neighborhood structure in robot path planning and perception often forms a substitution tiling, in which the same structure appears on multiple scales [11, 22, 23]. ADAPT searches for this self-similar structure, and uses it to create schemas that function on many scales.

Self-similar structure appears in motion, perception, communication and planning. For example, the motions of a robot on a small hexagonal grid [22, 23] can be computed and analyzed, and then copies of the small grid can be composed in such a way that the properties scale up. In particular, the decomposition of motion planning into translations and rotations is found on the small grid, and this decomposition then scales up, so that it is known to hold on all scales on all hexagonal grids.

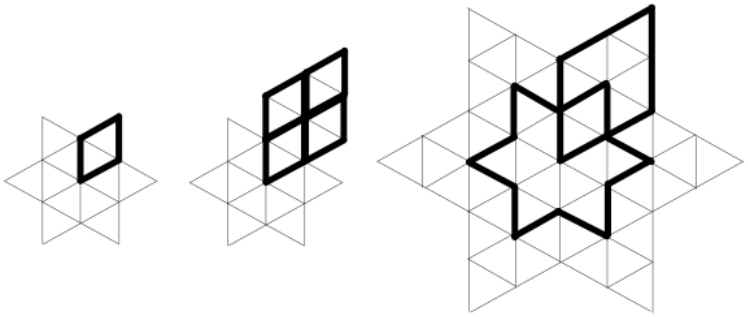

Figure 3. Composing hexagonal grids to create larger grids.

At left in Figure 3 is the "flower" grid. At center, we begin to extend by composing the upper right bold grid in all ways with itself. At right, this process is complete, giving a grid similar to the original grid.

The structure of local neighborhoods often replicates on larger scales. Features scale up, with features of the basic neighborhood appearing in larger neighborhoods at larger scales, e.g. the larger hexagons that are composed from smaller hexagons. By planning locally within neighborhoods of different scales, the robot can follow a very complex path without performing any complex planning; the complexity of the path reflects the complexity of the environment rather than the complexity of the planner. In Figure 4, the robot acts at different scales, avoiding obstacles while moving towards G. The resulting complex, zigzag path seemingly exhibits intricately planned behavior.

Perception exhibits this self-similar structure, too. Barnsley [6] has shown that this 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. A number of research and commercial efforts are underway in this area, including those at the University of Waterloo, INRIA and Ecole Polytechnique de l'Universite de Montreal.

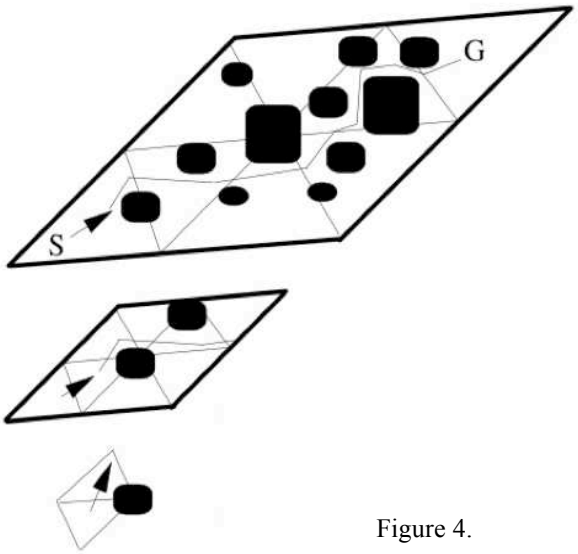

Figure 4.

Thus, this approach integrates perception and action in a principled way by connecting their local symmetries and invariants. 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 task and the environment. When such a structure exists, the robot can exploit it to solve large, complex problems in an efficient hierarchical manner. RS/Soar provides a perfect environment in which to use this approach to synthesis.

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

A 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.

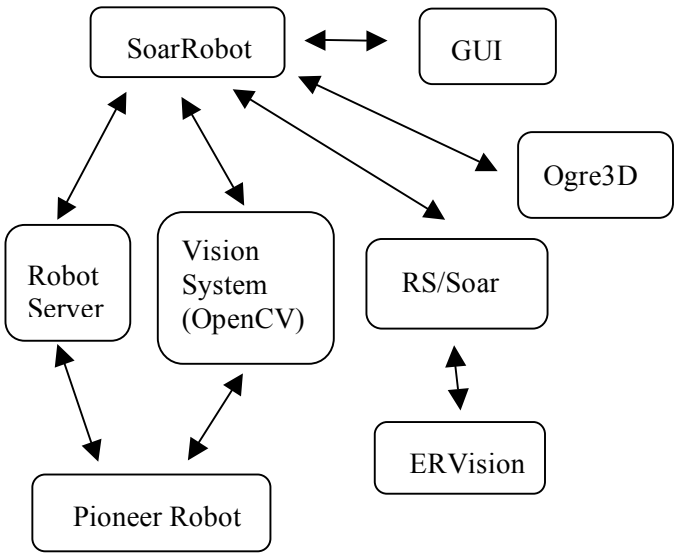

Figure 5. ADAPT's structure.

Figure 5 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 Robots. 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 (every tenth of a second), 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 Figure 5 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.

The entire system is held together by the SoarRobot program, which is written in Java. SoarRobot coordinates the interaction of all components and the user (via the GUI), and runs on the base PC, which is a standalone workstation.

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, currently under construction, will use this visualization capability in the same way that NL-Soar understands the world using "comprehension through generation" [34]. 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.

3.1. The Predictive Vision System

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 controlled by RS/Soar.

The bottom-up component component is designed to be simple and fast. It does this by not producing much detail. The idea is for it to produce a basic stereo disparity map, a coarse-grained image flow, and color segmentation in real time. It runs on the robot's onboard computer using Intel's open vision library, and segments the visual data from the robot's two framegrabbers. These "blobs" are transmitted together with stereo disparity data and optical flow to the offboard PC that is running RS/Soar, where it is placed into working memory. This component is always on, and its output is task-independent.

The top-down component executes the more expensive image processing functions, such as object recognition, sophisticated image flow analysis, and application of particular filters to the data. These functions are called in a task-dependent and goal-dependent manner by RS/Soar operators. This greatly reduces their frequency of application and speeds the operation of the vision system significantly.

These two components are not connected to each other; instead, the output of the bottom-up component is used by RS/Soar to determine when to call the top-down operations. RS/Soar compares the bottom-up output to the visual data predicted by the mental model, as illustrated in Figure 6.

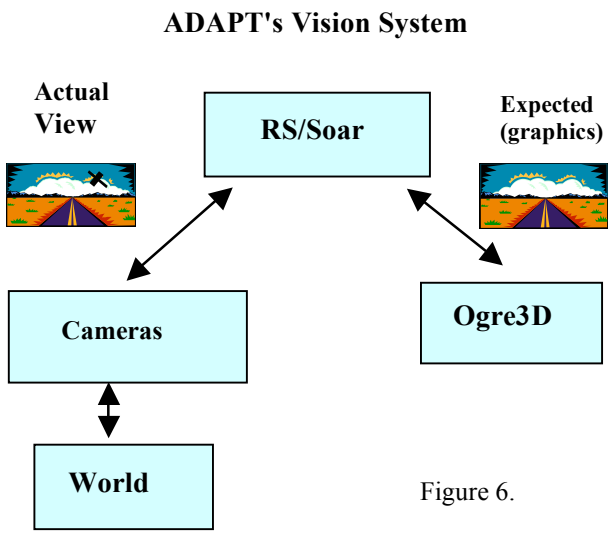

Figure 6.

The mental model can display the view that the virtual copy of the robot "sees" in the virtual environment. The output of this graphics "camera" in Ogre is segmented and placed into RS/Soar's working memory, together with distance information and motion information from the mental model. Soar operators test for significant differences between the expected view and the actual view, e.g. the appearance of a large new blob or a large change in optical flow. Any significant difference causes an operator to be proposed to attend to this difference.

For example, if a new blob appears, an operator will 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 RS/Soar will instruct the robot to turn its cameras towards this blob, and then call its recognition software to process the rectangular region of the visual field that contains the blob. The recognition software currently used is ERVision 3.0.

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 approximately match the expected blobs from Ogre, ADAPT assumes it is the same object. Recognition becomes an explicitly goal-directed process that is much cheaper than continually recognizing everything in the environment. The frequency with which

these expensive operations are called is reduced, and they are called on small regions in the visual field rather than on the whole visual field.

Thus, ADAPT's vision system spends most of its time verifying hypotheses about its environment, instead of creating them. The percentage of its time that it must spend attending to environmental changes depends on the dynamic nature of the environment; in a relatively static environment (or one that the robot knows well from experience) there are very few unexpected visual events to be processed, so visual processing operators occupy very little of the robot's time.

Another advantage of this approach to visual comprehension is that it opens the possibility of learning recognition strategies.

3.2. Language Semantics from Image Schemas

Another important use of this world model is to enable the robot to "see" what utterances might mean, and thus to help select appropriate semantics from among numerous possibilities. Langacker's Cognitive Grammar [43, 44, 45] provides a well-founded integration of grammar and semantics with imagery, using spatial primitives to give semantics for many common actions and relationships. His grammar provides a mechanism for reasoning about linguistic composition by superimposition of images. For example, Figure 7 shows image schemas for "walk" and "John" and "snake". Given the sentence, "John walks", the schema for "walk" can be completely assigned to "John". But when given the sentence, "A snake walks", the schema for "walk" cannot be completely assigned to the schema for "snake". In this way, the system can figure out that the first sentence makes sense and the second one doesn't.

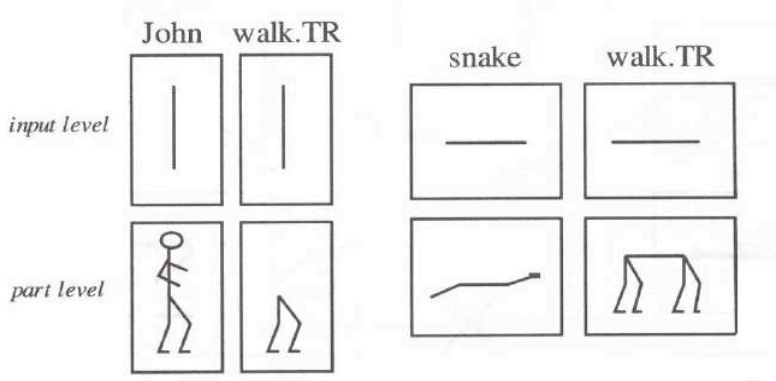

Figure 7. Image schemas for "John walks" and "A snake walks" (from Holmqvist, 1999).

Holmqvist [36, 37] partially implemented this grammar, but no complete implementation yet exists. We are currently implementing Langacker's Grammar within NL-Soar, using the Ogre world model to represent image schemas, and will extend his grammar by animating these schemas, so that the "walk" schema will not be a static picture of legs, but rather a working model of virtual legs. Soar will search for ways to animate image schemas successfully in the world model, and will chunk solutions to speed future comprehension.

In this way, perceptual patterns from the vision system are used not only to guide motion, but also to guide ADAPT's search among alternative semantics for utterances, both the system's own and those it hears. This is an illustration of the deeper integration of perception, language and action in ADAPT.

This part of the project is new, having started only in the past few months.

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. Since our robots possess simple grippers instead of flexible arms with grippers, we cannot pose grasping tasks.

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. In particular, we can embed a number of well-tested cognitive tasks in this class, e.g. the Towers of Hanoi, as illustrated below. This permits us to compare the robot's performance and learning curves with human data. Although we do not expect them to match, we wish very much to be able to make this comparison to explore the similarities and differences between human and robot cognition.

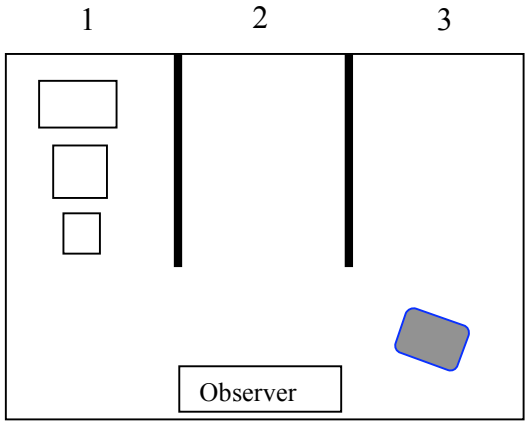

Figure 8. The robot (gray) must move the three boxes from the leftmost column to the rightmost. It can push or lift one box at a time. A box can never be in front of a smaller box, as seen from the observer. The larger a box is, the taller also. The front area must be kept clear so the robot can move; there can be no boxes left in this area.

A schema 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".

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.

5. Plan of Work and Coordination Plan

This project brings three existing, mature research efforts into an interdisciplinary collaboration based on shared methodologies and assumptions. Each of the three principal investigators brings an expertise in one of the three cognitive abilities that are the focus of this project. Prof. Lonsdale is an expert in natural language processing. Prof. Lyons has a great deal of knowledge in robotics and vision. Prof. Benjamin focuses on problem solving and reformulation. Professors Lyons and Benjamin are both in New York City, they and their students interact frequently. Prof. Lonsdale is developing NL-Soar at BYU. Regular code updates are sent to Pace University's Lab and Fordham University's Lab to be tested on the robots. There are more than a dozen graduate and undergraduate students involved in the project currently.

One of the broader impacts of this project is the establishment of this interdisciplinary, multi-university collaboration. There is a need for more interaction between the fields of linguistics and robotics. Also, this project has led to the creation of a new graduate course at Pace University. This course, CS630 - Intelligent Agents, teaches Soar and embodied cognition, and has already led to a spin-

off collaboration with the curator of the South Street Seaport Museum in New York City on data mining of historical databases.

To date, the following has been achieved:

- The integration of RS and Soar has been implemented. Soar can control the movements of the Pioneer robots and model and predict their motion successfully in the virtual world.
- Soar can create basic entities in the virtual world based on its vision data and update them to reflect new percepts as the robot moves.
- NL-Soar has been connected to the robot's speech recognition and speech production software. It has been demonstrated successfully that NL-Soar can listen to a person, generate an appropriate response using a discourse model, and speak the response.
- An initial version of the reformulation algorithm has been implemented and tested in Soar. It successfully finds self-similar patterns in puzzles and other simple AI tasks, as well as in motion planning tasks.

Currently, we are working on the following activities:

- Image schemas are in the design stage. Within a year they will be connected to the virtual world and used for language semantics.
- The mechanism for Soar to chunk over concurrent schemas is being developed.
- The repertoire of virtual objects that Soar can create and control is being expanded.
- The planner (including the virtual world visualizer) is being expanded.

A broader impact of this project is that exceptional undergraduate students are now involved in performing research, both at Brigham Young University and at Pace University. For example, one of these undergraduates is now programming the repertoire of objects in virtual world.

A further impact of this project is the inclusion of students who are members of underrepresented groups. Most of the Pace University students who have worked on ADAPT are members of groups that are underrepresented in the research community. In particular, most are female, and female researchers are participating in this project at all three universities.

Over the next three years, we envision the following major activities and milestones:

Year 1

- Expand NL-Soar's discourse model.
- *Milestone:* Report on the performance of NL-Soar with speech recognition and production.
- Finish designing and implementing the image schemas.
- *Milestone:* Report on the performance of NL-Soar using image schemas.
- Specify the vision, natural language and problem solving elements in the class of shepherding problems.
- Complete the design of the schema language for perception, language, and problem solving.
- *Milestone:* Demonstrate effective problem solving and chunking on simple shepherding tasks without language.

Year 2

- Integrate the complete schema language into the architecture, connecting it to all modalities and capabilities: vision, sonar, language, planning.
- *Milestone:* Demonstrate the learning (by chunking) of key visual percepts in shepherding problems. These percepts will include depth layers, barriers and openings, and object continuity.

Year 3

- *Milestone:* Demonstrate the ability to communicate task knowledge to the robot in English and explanations from the robot in English while it is solving simple shepherding tasks.
- Identify and evaluate instances of useful cross-area pattern recognition.
- Evaluate use of cross-area pattern recognition by the architecture.

- Conduct a preliminary study to determine the benefits of cross-area learning in other problem domains.
- *Milestone:* Demonstrate effective problem solving and chunking on complex shepherding tasks.

5.1. Deliverables

- A common hierarchical schema-based representation framework for the areas of vision, language and problem solving. This deliverable will be in the form of research publications.
- Demonstrations of the system learning and applying patterns separately in the areas of vision, natural language and problem solving. The demonstration will consist of one or more tasks from the class of shepherding problems described earlier. The demonstration will use the implementation describe in 3 and will be shown at Pace's Robotics Lab.
- Demonstration of a pattern learned in one area applied to another. Again, the demonstration will be one or more tasks from the class of shepherding problems. The demonstration will use the implementation describe in 3 and will be shown at Pace's Robotics Lab.
- An evaluation of the benefits of the proposed cross area pattern learning for the class of shepherding problems, and a preliminary study of the impact of cross area pattern learning on other problem types. This deliverable will be in the form of research papers.

The NL-Soar component, the predictive vision system, the Soar reformulation/planning component and the RS/Soar implementation will be developed as standalone components, to be usable by other robotics researchers individually, and will be made publicly downloadable. A broader impact of this project is the potential use of NL-Soar as a natural language interface for the disabled. A further impact is the availability of these separate components for use by other researchers.

In addition, the descriptions of the shepherding tasks and the robot's performance on them will be made public so that other researchers can test their robots on these same tasks and compare performance. All of the components will be fully documented in refereed publications.

5.2. Coordination Plan

The specific roles of the PI, Co-PIs, other Senior Personnel and paid consultants at all organizations involved:

There are three main investigators. Prof. Benjamin of Pace University is the PI who handles the organizational aspects of the ADAPT project. Prof. Lyons of Fordham U. and Prof. Lonsdale of BYU are co-PIs. Their roles in the project are as follows.

D. Paul Benjamin, Pace University is the Lead researcher into the working memory/virtual world subsystem, the method of synthesis by decomposition, and the overall structure and system implementation of ADAPT.

Damian Lyons, Fordham University is the Lead computer vision researcher, originator and developer of the RS language, and designs the predictive vision subsystem.

Deryle Lonsdale, BYU is the Lead natural language researcher and head of the NL-Soar project (the natural language subsystem). He is designing the image schemas and is integrating them into NL-Soar.

How the project will be managed across institutions and disciplines:

We currently coordinate the project using frequent videoconference or audioconference discussions, several meetings, and a software repository. The project is two years old, and we have had success with this mix. We feel this is because this project factors pretty cleanly into three separate pieces corresponding to the disciplines of the three professors. Prof. Lyons works on the visual system and its connections to the virtual world. Prof. Lonsdale works on the natural language system. Prof. Benjamin works on the synthesis (planning and problem solving) and learning. Each of these components can be advanced separately for months at a time. When a new capability has been ironed out, it is entered in the software repository and communicated with the other research groups.

This loose form of collaboration has proved successful for us. It lets the three researchers follow their

individual instincts. We share a common research perspective that is reinforced by our frequent discussions, so that our three subprojects remain compatible. It would be better if we did have an annual workshop that brought us all together, but we do not have funds for that (the request is in the budget).

Pace University and Fordham University have the same robot models (Pioneer P2 robots with stereo color vision), which makes it easy to transfer software and to compare results. Currently, Prof. Lonsdale at BYU has no robot (linguistics professors rarely do) and we are approaching the point when he will need to have one to continue to make consistent progress. Part of the equipment budget is for the purchase of a new robot to replace an older one at Pace University, which will then be moved to BYU.

Identification of the specific coordination mechanisms that will enable cross-institution and/or cross-discipline scientific integration (e.g., yearly workshops, graduate student exchange, project meetings at conferences, use of the grid for videoconferences, software repositories, etc.):

The project is currently coordinated by frequent videoconferences and audioconferences, together with local meetings and a shared software repository. We hold videoconferences or audioconferences approximately every three weeks. During these meetings, we discuss implementation issues and also basic theoretical concerns. We share research papers that we have read and discuss their implications for our project. Part of the equipment budget in the first year is to improve our videoconference equipment.

Due to the proximity of Pace University and Fordham University, the researchers at these sites are able to meet whenever necessary. Typically, these meetings will consist of Profs. Lyons and Benjamin, together with several graduate students. Several such meetings have taken place over the past year.

We use a software repository to coordinate the software for the project. It is located at https://robotlab.csis.pace.edu/robotlab/ and consists of a subversion repository, together with a wiki.

We would very much like to hold an annual workshop for all the professors and grad students, together with a few invitees whose work is influencing our project. We do not have funds now for anything on that scale, and have included it in the proposal budget.

Specific references to the budget line items that support these coordination mechanisms:

Videoconference upgrades are included in the Materials and supplies item G.1. Travel (item E.1) includes travel to a yearly project workshop. The Equipment item in year 2 (item D.1) includes the cost of a new robot. This new robot will possess an arm and gripper so that we can expand the class of tasks to include grasping, which is a significant expansion. Also, this will enable us to move an existing robot to BYU so that the linguistics portion of the project can have one to fully test NL-Soar.

6. 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.