A Cognitive Robotics Approach to Comprehending Human Language and Behaviors

D. Paul Benjamin
Pace University
1 Pace Plaza
New York, New York 10038
001-212-346-1012
benjamin@pace.edu

Deryle Lonsdale
Brigham Young University
Dept. Linguistics & English Language
Provo, Utah 84602
001-801-422-4067
lonz@byu.edu

Damian Lyons
Fordham University
340 JMH, 441 E. Fordham Rd.
Bronx, NY 10458
001-718-817-4485
dlyons@fordham.edu

ABSTRACT

The ADAPT project is a collaboration of researchers in linguistics, robotics and artificial intelligence at three universities. We are building a complete robotic cognitive architecture for a mobile robot designed to interact with humans in a range of environments, and which uses natural language and models human behavior. This paper concentrates on the HRI aspects of ADAPT, and especially on how ADAPT models and interacts with humans.

Categories and Subject Descriptors
I.2.9 [**Artificial Intelligence**]: Robotics – *autonomous vehicles.*

General Terms
Performance, Design, Experimentation.

Keywords
Cognitive robotics, learning, natural language

1. INTRODUCTION

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 linguists and 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 and use of natural language and speech generation.

ADAPT models the world as a network of concurrent schemas, and models perception as a problem solving activity. Schemas are represented using the RS (Robot Schemas) language [19,20], and are activated by spreading activation. RS provides a powerful language for distributed control of concurrent processes. Also, the formal semantics of RS [30] 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 Carnegie-Mellon University 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.

This cognitive robotic architecture is currently being developed on four Pioneer robots in the Pace University Robotics Lab and in the Fordham University Robotics Lab.

In the next section, we give a brief description of the structure and implementation of ADAPT to provide background for the rest of the paper. In the subsequent sections, we focus on ADAPT's natural language system and on ADAPT's method of comprehending its environment and other agents.

For more information on ADAPT, and especially on the vision system, see [3, 4, 5, 6]. More papers and videos on ADAPT are available at the Pace University Robotics Lab website.

2. THE ADAPT ARCHITECTURE

Our approach is fundamentally different from other projects, 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 [19,20], which provides a model for building and reasoning about sensory-motor schemas, with Soar [10], 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 [18]. Soar is a unified cognitive architecture [25] that has been successfully applied to a wide range of tasks.

Soar's model of problem solving utilizes a single mechanism of subgoaling and chunking to explain human problem solving performance; utilizing Soar as the basis of ADAPT permits us to unify the mechanisms underlying perception, language and planning. Furthermore, it permits us to explore possible interrelationships between learning in these areas, e.g. how learning language and learning perception may be related. Finally, it permits us to test our architecture on robotic versions of well-known cognitive tasks (explained in Section 5) and explore how robot learning might be related to human learning.

2.1 RS and Soar

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 sensory-motor, encompassing both perceptual and motor aspects,

Perception is active, goal-directed and task-dependent,

Control is distributed and concurrent, with a high degree of parallelism.

Even "pure" vision is sensory-motor, 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.

To this end, ADAPT integrates the Soar cognitive architecture and the RS (Robot Schemas) language. This is a straightforward way to try to build an architecture that can handle the entire abstraction hierarchy smoothly.

The RS (Robot Schemas) language is a language with a formal model of concurrent robot computation that is based on the semantics of networks of port automata. A port automaton (PA) is a finite-state automaton equipped with a set of synchronous communication ports.

RS builds a network of *sensory-motor schemas* to model the dynamics of both the robot and the environment. A schema is an organized network of actions (abstract and/or concrete), percepts, words, facts and beliefs about some aspect of the world. Schemas also contain explicit qualitative temporal information, e.g. about intervals of time during which an action must take place. The schema relates these components to each other and to other schemas.

For example, the schema for a table would contain various images of tables and words that describe tables, and would connect them to facts about tables, such as their typical size, and to actions that typically are performed with tables, such as sitting and eating at one (which is another schema). The table schema would be connected to the schemas for chairs, dining rooms, conference rooms and many other relevant concepts.

One of the unique advantages of RS is its formal semantics. Each schema has an associated port automaton that defines the semantics of the schema. ADAPT explicitly constructs this PA for each schema and attaches it to the schema. The PA provides the basic semantics for the use of natural language. States in the PA provide the semantics for relations and predicates (adverbs and adjectives). Transitions between the states provide the semantics for verbs.

A large body of work in intelligent systems employs similar notions of schema. Roy's work [26] focuses mainly on the integration of anguage with action. Arbib [2] presents a comprehensive theory of schemas.

ADAPT uses schemas by instantiating them to create schema instances that are connected to each other to form a network. Over time, ADAPT maintains this network, monitoring its behavior and adding schema instances and removing old ones as its goals and the environment change.

RS has been used successfully in industrial settings for applications such as kit assembly [18]. RS thus provides a mature language for robust, real-time distributed control. A video is available at http://csis.pace.edu/robotlab/clips/puma.avi that shows RS controlling a robot arm and camera to track and grasp an erratically moving object.

RS provides a powerful representational language for the system's dynamics, language and percepts; however, RS does not provide a mechanism for synthesizing the dynamics. Furthermore, RS lacks cognitive plausibility, and in particular lacks a learning method.

We have implemented RS in Soar [10] to take advantage of Soar's cognitively plausible problem-solving and learning mechanisms, as well as the existing work on other aspects of cognition in Soar, including natural language [13,27], concept learning [31], and emotion [22].

Soar uses *universal subgoaling* to organize its problem solving process into a hierarchy of subgoals, and uses *chunking* to speed and generalize that process. ADAPT uses Soar's subgoaling ability to create hierarchies of schemas, and uses Soar's chunking ability to speed the creation of these hierarchies. One of the principal goals of the ADAPT project is to investigate how this form of learning can speed scene comprehension and language learning.

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.

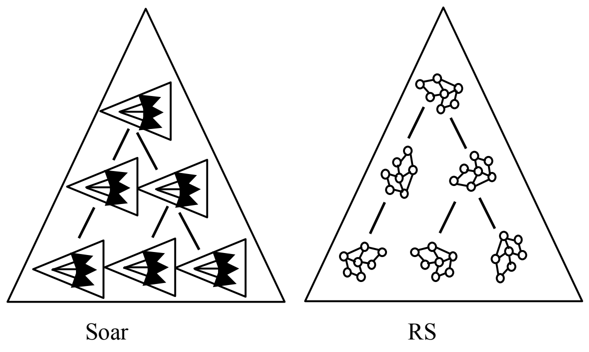


Figure 1. Soar problem spaces implement RS port automata.

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. Its weaknesses are the lack of a synthesis 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 have created an architecture to:

- 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.

2.2 Using a Mental Model to Predict Behavior

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. Ogre gives the robot the ability to create a dynamic virtual model of its environment by providing sophisticated graphics and rendering capabilities together with a physics engine based on the Open Dynamics Engine. Ogre is capable of modeling other agents moving and acting in those environments.

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 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.

ADAPT uses Ogre to build a qualitative, approximate version of its environment. Not all objects in the environment are rendered; only those relevant to current tasks are rendered. The precision of the initial rendering is low. If more precision is needed for an object, ADAPT must spend more time perceiving that object, possibly including moving around the object to gather more data about it. In practice, the approximate model created by ADAPT is sufficient for path planning.

Figure 2 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 Java program that interfaces with the robot's hardware. The server program listens for a call from SoarRobot (every tenth of a second), then returns all the robot's data (including the sonar data) back to SoarRobot. While the server is waiting, the Vision System runs on the robot's PC.

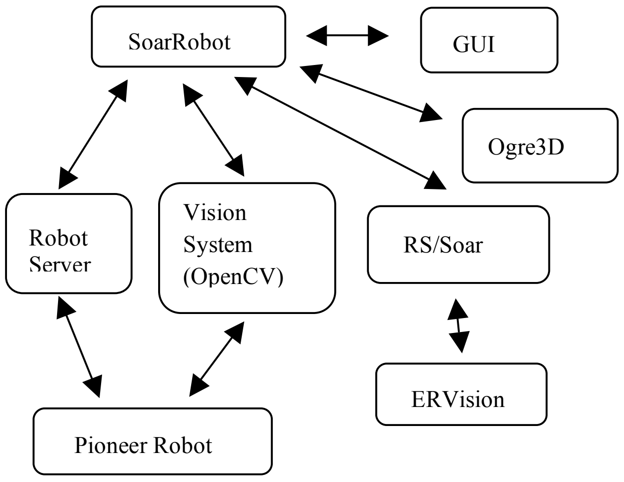


Figure 2. ADAPT’s structure.

The lower right part of Figure 2 shows the components that run on the base PC. These include the 3D world model, which is the Ogre3D gaming engine, and Soar, which contains an implementation of RS schemas. Soar also can call the ERVision object recognition software.

The GUI is 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 Ogre’s physics plugin, the robot can examine how the world might evolve in the near future. ADAPT's planner uses this visualization capability in the same way that NL-Soar understands the world using "comprehension through generation" [24]. 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.

2.3 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 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.

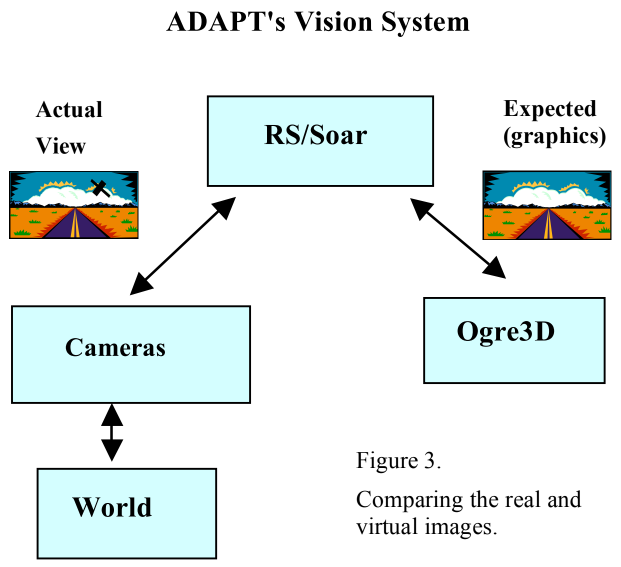


Figure 3.

Comparing the real and virtual images.

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.

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.

2.4 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 [11,12] 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 4 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 more sense than the second one.

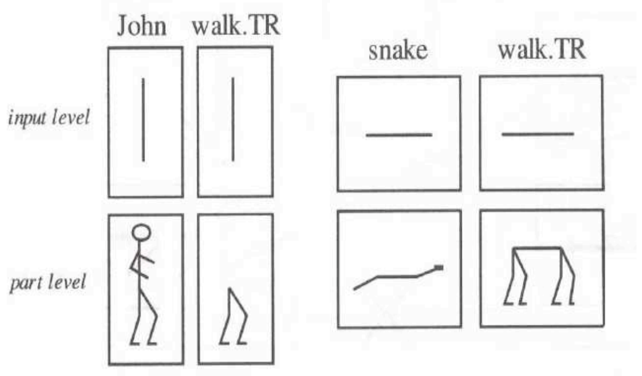


Figure 4. Image schemas for "John walks" and "A snake walks" (from Holmqvist, 1999).

Holmqvist [8,9] 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.

3. ADAPT's Natural Language Component

Communication between humans and the robot is handled by NL-Soar, which is a natural language system implemented within the Soar cognitive modeling framework [13-17]. The system supports spoken human language input via an interface with Sphinx, a popular speech recognition engine. 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 to understand the text:

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

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 [21], exemplar-based [38], and hybrid [17] 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 [20].

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

As is typically implemented for human/robotic interaction, our system uses a 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: (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. 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, but still are not as flexible as highly dynamic environments may require. BDI architectures 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.

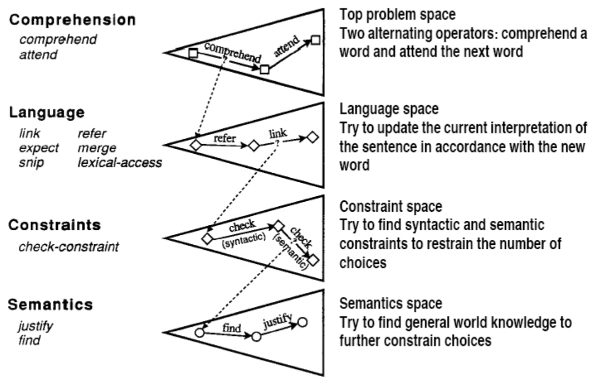
NL-Soar implements a discourse recipe-based model (DRM) for dialogue comprehension and generation [24]. It learns the discourse recipes, which are generalizations of an agent's discourse plans, as a side effect of dialogue planning. This way, plans can be used for comprehension and generation. If no 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.

4. Comprehension Through Generation

NL-Soar implements a discourse recipe-based model for dialogue comprehension and generation. 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.

This approach to comprehension is based on the work of Green & Lehman [7], who used Soar to model discourse planning in natural language. Their goal was a computational model of discourse, in which Soar would interact with a human using English. In their task, Soar had to do two things: construct an explanation of an incoming utterance and generate an appropriate response. Their approach was to use ***comprehension through generation***, in which Soar would construct an explanation for the incoming utterance by attempting to generate a similar utterance on its own. Soar would search among possible combinations of goals and expectations to try to generate this utterance, and when it succeeded in matching the utterance it assumed the human's goals and expectations were those it had found. This gave Soar an understanding of the goals of the person it was conversing with, so that Soar could choose appropriate goals for generating a response. Soar also formed a chunk for this search, so that a similar utterance in the future would be understood in one step.



Comprehension
comprehend
attend
comprehend
attend
Top problem space
Two alternating operators: comprehend a word and attend the next word
Language
link
refer
expect
merge
snip
lexical-access
refer
link
Language space
Try to update the current interpretation of the sentence in accordance with the new word
Constraints
check-constraint
check (syntactic)
check (semantic)
Constraint space
Try to find syntactic and semantic constraints to restrain the number of choices
Semantics
justify
find
find
justify
Semantics space
Try to find general world knowledge to further constrain choices

Figure 5. Example goal stack in NLSoar [7].

For example, if the discourse were about cooking and the incoming utterance were, "Turn the flame down." then Soar would try to generate that same sentence by searching various combinations of goals and knowledge. One combination that works is the knowledge that a high flame cooks food faster and the goal is to cook it slowly. If Soar found this combination, then it would assume the speaker has this same combination of knowledge and goal. This approach requires Soar to have knowledge about cooking, so that it can search that knowledge.

Comprehension through generation is used by ADAPT to understand incoming utterances, just as it is by NLSoar. But ADAPT uses comprehension through generation not only for speech input, but also to understand nonverbal behaviors of other agents in its environment.

Nonverbal behaviors are extremely important to understand, because much of the interaction between agents in the real world is nonverbal. For example, walking alongside a person involves keeping a relatively constant distance from the person, which requires monitoring the person's movements and responding appropriately to them. If the person speeds up or slows down or changes direction, there is probably a reason, and an intelligent robot should understand that reason in order to respond appropriately. If a team is composed of several people and several robots, it is not possible for the people to continually explain their movements to the robots; it is necessary for the robots to be able to generate their own explanations.

ADAPT generates these explanations by using the Ogre3D world model to simulate the observed behaviors of other agents. Each agent is represented as a Soar agent that is identical to the robot. ADAPT attempts to replicate the observed behavior by searching through sequences of Soar operator firings for that agent. When an appropriate sequence of Soar operators is found that replicates the observed behavior, ADAPT assumes that the agent has executed this sequence of operators, and attributes to the agent the goals and knowledge used by these operators.

For example, suppose the robot is traveling with a human teammate and the observed behavior is that the human looks ahead and slows down a bit because he sees a vehicle crossing his path and wishes to let it pass. This is a typical behavior that should not need to be explained to the robot. ADAPT can model the human and the vehicle in its 3D world model, and "imagine" the consequences of not slowing down, which will be a collision. This permits ADAPT to infer the reason for the behavior, and to create a new schema for slowing down to let vehicles pass. This schema is used both for generation (when the robot needs to slow down to permit vehicles to pass) and for comprehension.

The advantage of comprehension through generation is that it provides a deeper understanding of the world than statistical associations can provide. In particular, ADAPT can reason about why complex sequences of actions are performed, and how they can be varied. The disadvantage of this approach is that it is slow; the search for the proper combination of goals and knowledge can be extremely long, and may not be successful. The usefulness of this approach to robotics, and especially to HRI, is not known yet, and will depend on the power of Soar's chunking method.

5. Task Definition for Evaluating ADAPT

The goal of the ADAPT architecture is to increase the robot's comprehension of its environment, and especially of the humans in 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, a team consisting of one or more mobile robots and people have the objective of moving one or more objects so that they are grouped according to a given constraint. An example 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, navigational and communication difficulties, rather than just as abstract tasks.

5.1 A Simple Example

This is an example of a shepherding task.

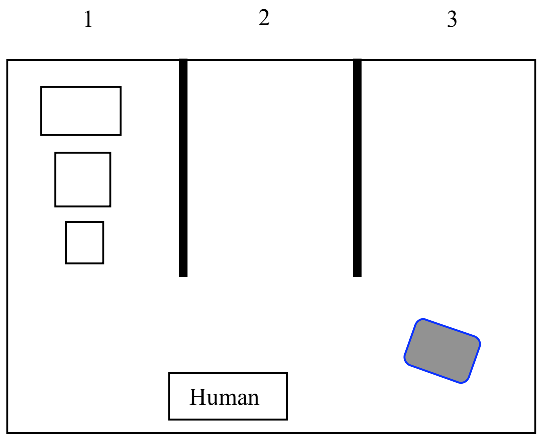


Figure 6. A robotic version of the Towers of Hanoi.

The robot (gray) must cooperate with a human to move the three boxes from the leftmost column to the rightmost. The robot can push or lift one box at a time. A box can never be in front of a smaller box, as seen from the front of the room. 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" or “wait for the human to move the small box”.

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.

6. Summary

We have sketched the overall design of a new robotic cognitive architecture. The goal of ADAPT is to integrate visual comprehension, use of natural language, problem solving and learning in an architecture that is embedded in the physical world. The design of the visual system reflects this goal by using a sophisticated world model to comprehend and predict the behavior of the environment.

The implementation of the basic components of ADAPT is complete, including the schemas in Soar, the use of Ogre3D to create world models, and the predictive vision system. Experiments to evaluate its performance and learning are currently under way. NL-Soar and the image schemas are currently being developed separately at BYU using a copy of ADAPT connected to a simulated robot.

Our current HRI focus is on developing and implementing more sophisticated discourse models, and on better integration of the semantics of RS schemas with NL-Soar semantics.

7. 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] Michael A. Arbib, 2003. "Schema theory", in Michael A. Arbib, editor, The Handbook of Brain Theory and Neural Networks, 2nd Edition, pages 993–998. MIT Press.

[3] Benjamin, D. Paul, Damian Lyons and Deryle Lonsdale, “Embodying a Cognitive Model in a Mobile Robot”, Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision, Boston, October, 2006

[4] Benjamin, D. Paul, Tom Achtemichuk and Damian Lyons, “Obstacle Avoidance using Predictive Vision based on a Dynamic 3D World Model”, Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision, Boston, October, 2006.

[5] 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.

[6] 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.

[7] Green, N. and Lehman, J.F., “An Integrated Discourse Recipe-Based Model for Task-Oriented Dialogue.” *Discourse Processes, 33*(2), 2002.

[8] Holmqvist, K., “Implementing Cognitive Semantics”, Lund: Department of Cognitive Science, 1993.

[9] Holmqvist, K., “Conceptual Engineering”, in Cognitive Semantics: Meaning and Cognition, J. Allwood and P. Gardenfors (Eds.), John Benjamins, pp. 153-171, 1999.

[10] Laird, J.E., Newell, A. and Rosenbloom, P.S., “Soar: An Architecture for General Intelligence”, *Artificial Intelligence* **33**, pp.1-64, 1987.

[11] Langacker, R., “Foundations of Cognitive Grammar, Vol. I”, Stanford University Press, 1987.

[12] Langacker, R., “Foundations of Cognitive Grammar, Vol. II”, Stanford University Press, 1991.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] Lyons, D.M., "Representing and Analysing Action Plans as Networks of Concurrent Processes", IEEE Transactions on Robotics and Automation, June 1993.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Newell, Allen, *Unified Theories of Cognition*, Harvard University Press, Cambridge, Massachusetts, 1990.

[26] Roy, Deb and Alex Pentland, 2002. "Learning words from sights and sounds: A computational model", Cognitive Science, 26(1):113–146.

[27] 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.

[38] 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.

[29] 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.

[30] Martha Steenstrup, Michael A. Arbib, Ernest G. Manes, *Port Automata and the Algebra of Concurrent Processes*. JCSS 27(1): 29-50, 1983.

[31] 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.