Designing a Robot Cognitive Architecture with Concurrency and Active Perception

D. Paul Benjamin

Pace University Robotics Laboratory and Computer Science Department
1 Pace Plaza, New York, New York 10038
benjamin@pace.edu

Deryle Lonsdale

Brigham Young University Department of Linguistics and English Language
Provo, Utah 84602
lonz@byu.edu

Damian Lyons

Fordham University Robotics Laboratory and Department of Computer & Information Science
340 JMH, Fordham University, 441 E. Fordham Rd., Bronx, New York 10458
dlyons@fordham.edu

Abstract

We are implementing ADAPT, a cognitive architecture for a Pioneer mobile robot, to give the robot the full range of cognitive abilities including perception, use of natural language, learning and the ability to solve complex problems. Our perspective is that an architecture based on a unified theory of robot cognition has the best chance of attaining human-level performance.

Existing work in cognitive modeling has accomplished much in the construction of such unified cognitive architectures in areas other than robotics; however, there are major respects in which these architectures are inadequate for robot cognition. This paper examines two major inadequacies of current cognitive architectures for robotics: the absence of support for true concurrency and for active perception.

The ADAPT architecture addresses these issues by integrating three theories: the theory of cognition embodied in the Soar system, the RS formal model of concurrent sensorimotor activity and an algebraic theory of decomposition and reformulation These three component theories have been implemented and tested separately and their integration is currently underway. This paper describes these components and the plan for their integration.

Introduction

The current generation of behavior-based robots is programmed directly for each task. The robot agents are written in a way that uses as few built-in cognitive assumptions as possible, and as much sensory information as possible, in a manner similar to Brooks (1986), who proposed in the mid 1980's an approach that emphasized fast, reactive actions and the absence of explicit models. 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 (Nicolescu & Mataric, 2000). Complex tasks that involve reasoning about spatial and temporal relationships require robots to possess more advanced mechanisms for planning, reasoning, learning and representation.

One approach to this issue is to equip a robot agent with a behavior-based "reactive" module coupled to a "deliberative" module that engages in cognitive reasoning – these are called hybrid reactive/deliberative systems (Arkin, 1998)(Lyons & Hendriks, 1991). Many such systems consist of a symbolic planner, or planner and scheduler, as the deliberative component. The planner formulates long-term and more abstract tasks, while the behavior-based system carries out the steps in the task in a robust fashion (Arkin, 1998). For example, Lyons and Hendriks (1995a, 1995b) describe a hybrid system based on the concept of iterative and forced relaxation of assumptions about the environment. The reactive system is implemented using the port-automata based RS model (Lyons, 1993) and ITL (Allen, 1981) 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 whose job is to report back 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 then respond to the violation of assumptions about the environment by re-planning. This hybrid approach endows a robot with a more sophisticated planning ability than either deliberative planning or reactive behavior provides alone; however, it does not address either learning or representation issues.

Furthermore, a planner can only consider plans based on the action and sensing repertoire it possesses. In this sense, it is as dependent for robust behavior on its designer as is a behavior-based system. The planner is constrained by the environment built in by its designer (or that it learned.) Each formulation makes certain results easy to obtain and makes others difficult. As a result of these two limitations, robots can usually do a few things very well, and everything else poorly.

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. 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 Garcia-Martinez & Borrajo (2000), Long, Fox & Hamdi (2002), and Veloso et al. 1995.

Our approach is *not* to glue together various modules to add capabilities to the system, but rather to build upon a unified architecture with more general cognitive abilities.

The cognitive science community has developed a number of unified cognitive models, including EPIC (Kieras, Wood & Meyer, 1997), Soar (Laird, Newell & Rosenbloom, 1987), and ACT-R (Anderson, 1996). These models contain comprehensive theories of cognitive structures based on decades of psychological experimentation, and were developed specifically to model human performance on a wide range of tasks. The models were evaluated based on the degree to which they fit human performance data on these tasks, especially timing data. These tasks typically were solving puzzles of various types and performing simple motor tasks, or learning basic cognitive tasks such as word recognition or sentence disambiguation. The model that incorporates a detailed model of perception (EPIC) faithfully reproduces the restrictions and timings of human perception (particularly eyes and ears). Other architectures such as Soar and ACT-R have recently borrowed this model from EPIC.

These unified cognitive models have a great deal to offer robotics. They are capable of sophisticated behaviors on a wide range of tasks, and exhibit human-like problem solving and learning. It is very plausible that the cognitive structures developed in these programs could enable similarly sophisticated behaviors by robots. However, the cognitive science community has not focused primarily on issues of sensory and motor processing, but rather on higher-level cognition. As a result, the sensory and motor parts of these architectures are not as well developed. These architectures have very detailed models of problem solving and learning, but comparatively sketchy models of how sensory processing and motor control are actually done and are interconnected.

In particular, these models lack two important features of most modern robot control programs: structures for truly concurrent processes, and structures that support active perception.

The goal of the ADAPT project is to build a complete cognitive robotic architecture by integrating RS (Lyons, 1993)(Lyons & Arbib, 1989), which provides a model for building and reasoning about concurrent, real-time sensory-motor schemas, with Soar. RS possesses a sophisticated formal language for reasoning about networks of port automata and has been successfully applied to robot planning and factory automation (Lyons & Hendriks, 1995b). Soar is a unified cognitive architecture (Newell, 1990) that has exhibited successful problem solving and learning on a wide range of tasks.

This cognitive robotic agent is currently being implemented and tested on Pioneer P2 robots (http://robots.activmedia.com/).

The next two sections of this paper briefly describe the need for concurrency and active perception in robotics. The remainder of the paper gives an overview of the ADAPT architecture. First, we describe the use of schemas in ADAPT, and the use of reformulation as the means of generating behaviors. Then we give some details of search and learning in ADAPT, mainly to illustrate the use of structures from Soar and ACT-R. Finally, we describe the class of tasks we will use to test our robots.

Dynamics and Concurrency

Arbib (1993) makes the case that distributed computation is necessary for modeling of biological systems, and that this is so because distributed computation is necessary for intelligence. Whether this claim is true or not, it reflects the reality that many roboticists believe it and much current robotics research is based on distributed computational models. Many existing robot control paradigms consist of a hierarchy of behaviors, e.g. (Brooks, 1986) and (Lyons and Hendriks, 1995a) in which higher-level behaviors are not abstractions of lower-level behaviors, but instead affect their tendencies.

A robot typically possesses a large number of moving components, e.g. gripper joints, wheels or legs, pan-tilt camera units, camera lenses that zoom, and these components may all be in action simultaneously, in addition to many actions occurring simultaneously in the environment. In such a system, the central issue is not selection of an action, which is the principal focus of the cognitive modeling community, but rather the *coordination* of two or more actions to achieve a desired or observed behavior. This is true both of the robot's own actions and

of environmental actions the robot is modeling.

One particularly important instance of coordination is the coordination of sensory behaviors and motor behaviors. Robots do not tend to execute purely sensory acts or purely motor acts; real behaviors usually combine sensory and motor actions. For this reason, we usually refer to *sensory-motor* acts. For example, getting input from a camera requires continually adjusting the depth of field to bring the desired part of the environment into focus (this is true of the human eye, as well.) And motor acts should be coupled to relevant sensory input; when this is not done, robots get stuck in corners or against walls, or even on rocks on Mars!

The coordination of distributed programs is extremely difficult, so it is not surprising that cognitive modelers have avoided it. But it is necessary in robotics. The existing cognitive architectures permit parallel execution of actions, but do not provide any model of how concurrency should be modeled or controlled (and especially lack a model of how to learn concurrent actions.) This is in contrast to their detailed models of sequential search and learning of search control knowledge.

This lack is one of the major obstacles facing roboticists who wish to embody their work in one of the existing cognitive models, and is one of the main motivations for the development of ADAPT.

Active Perception

Much current robotics research models perception as an active process, rather than as a passive one. This means that perception is goal-directed and context-sensitive at every stage, including the initial processing of input sensory data. Active perception processes even low-level visual and auditory data in a goal-directed manner. For example, if a robot is crossing the street, it will not scan the entire visual field, but rather turn its head in the direction of oncoming traffic. It will not process all the parts of the resulting visual frame equally, but will filter the data for large blobs that are moving towards the robot, ignoring (or processing in a very coarse way) the rest of the visual field. Such a robot will perceive cars very differently depending on its goal and the situation.

This approach to perception greatly decreases the computational cost of perception because the system is applying only specific computational resources (such as visual filters) to chosen parts of the sensory data rather than applying all possible perceptual transformations to all parts of the data. Thus, active perception requires the robot to form sensory plans for gathering information from the environment, and transforms perception into a problem solving process. This permits the robot to apply its learning capabilities to perception.

Active perception, and in particular active vision (Blake & Yuille, 1992), is a major research area in robotics. As is the case with concurrency, existing cognitive architectures are not unable to utilize active perception but this is the exception rather than the rule.

To perceive actively, a robot cognitive architecture must have access to lower-level sensory and motor information. This is typically not done in current cognitive architectures.

The Rationale behind ADAPT

Need overall statement of philosophy of adapt: problem solving, reformulation, exploitation of self-similar structure, language -> soar, with act-r for language

ADAPT (**A**daptive **D**ynamics and **A**ctive **P**erception for **T**hought) is a cognitive architecture specifically designed for robotics. It is based on the Soar cognitive architecture and incorporates a formally precise model of concurrent processes and active perception. ADAPT has been under development for less than a year. This paper describes its design rationale, and the implementation as it exists.

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 permit researchers 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 that act at multiple scales to solve complex problems.

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.

The Structure of ADAPT

ADAPT resembles existing cognitive architectures in many ways, as it is based on one of them. It is a production-based architecture with a working memory, and

matches productions against working memory during each cycle in the same manner as these architectures. All the matching productions fire, as in Soar, and place their results in working memory. ADAPT possesses a long-term declarative memory, as ACT-R does, in which it stores general sensory-motor schemas. All perceptual processors fire in parallel, as in EPIC, but place their low-level sensory data into working memory, where it is accessible to the cognitive architecture's problem solving and learning mechanisms. Thus, ADAPT's overall organization resembles that of EPIC-Soar, with the additional property that general schemas are stored permanently in working memory.

The design of a robot cognitive architecture is faced with the need for multiple behaviors to be concurrently active. Yet, the robot must stay focused on its goal to achieve results in real time. In real-time situations, achieving a result too late is the same as not achieving it at all. If goals are allowed to proliferate, the robot will fail. We must keep the architecture from spending its time multiplying goals rather than solving them.

These two competing needs are met in ADAPT by drawing a distinction between the goals that are task goals, e.g. "find the blue block", and those that are goals of the architecture, e.g. "start the schema that scans the environment for a segment of a given color". Similarly, we distinguish between task actions, e.g. "pick up the block", and architectural actions, e.g. "start the gripper-closing schema".

We accomplish this dichotomy by partitioning the actions and goals into an architectural part consisting of architectural goals and actions and a task part consisting of task-specific goals and actions. We restrict the architectural part to one active goal (it has a goal stack) and one architectural action at a time. The architectural part is represented only procedurally, i.e. the system can execute the actions but cannot examine their internal representation. We represent the task goals and actions declaratively in working memory as well as procedurally, so that the robot can explain its actions as well as take instruction verbally. There can be as many active task goals and actions in working memory as the system wants. We call these goals and actions "schemas".

Schemas

The term "schema" has a long history in artificial intelligence, cognitive modeling and brain science, which we do not have space to recapitulate in full here. Minsky's frames are a type of schema, as are Schank's scripts. Arbib (1992) has developed a schema theory that essentially describes our use of the term. His schema theory is especially useful because it has been precisely specified with a formal semantics (Steenstrup, Arbib & Manes, 1983) and implemented in a language for real-time concurrent perception and action (Lyons & Arbib, 1989). We use this language, called RS (for Robot Schemas), to give our cognitive model the concurrent, real-time reasoning capability that it needs.

One very important aspect of schemas is that they combine procedural and declarative knowledge, i.e. all knowledge in a schema can be accessed declaratively for purposes such as communication and generalization, and accessed procedurally to solve problems. This distinguishes schemas from Soar operators or ACT-R declarative chunks. This combination of procedural and declarative representations is of central importance, as this permits behavioral knowledge to be both executed in procedural form and communicated by natural language in declarative form. Additionally, representing schemas in declarative form permits ADAPT to reason about and transform the hierarchy of schemas. For example, ADAPT will apply inductive inference operators to the declarative forms of schemas to generalize them.

Schemas are general patterns of perception and action, and are stored permanently in working memory. Each schema has a sensory part and a motor part, representing the goal of executing a particular motion depending on certain perceptual conditions. The motor parts of the schemas are the task actions. Schemas also contain explicit qualitative temporal information, e.g. about intervals of time during which an action must take place. General schemas are instantiated to create schema instances that are connected to each other to form a network.

The part of working memory that contains the library of schemas that can be instantiated and activated is similar to the declarative memory in which ACT-R 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 in a manner similar to ACT-R.

Schemas control the actual motors as a side effect of placing control information into working memory, which causes the robot to move. Since multiple schemas can be active simultaneously, parallel schema executions are possible (the RS language contains parallel combinators). It is the responsibility of the system (while it is planning) to ensure that simultaneous executions can be executed together. As only one architectural action is executable at a time, the architecture cannot start two schemas simultaneously or instantiate the variables of two schemas simultaneously. Specifically, it cannot initiate the processing of two different modes of sensory input at the exact same time. This respects the cognitive data available on dual tasking (Gopher & Donchin, 1986), so that compatibility is retained to some extent with human cognitive data.

ADAPT plans by transforming a hierarchy of schemas (there is a root schema "interact with the environment" to tie all the schemas together.) At each step, ADAPT can perform one of the following steps:

- refine a schema into subschemas,

- instantiate variables in a schema (this includes connecting two or more schemas by binding their variables together),
- start execution of a schema,
- suspend execution of a schema, or
- terminate and remove a schema.

Thus, ADAPT operators work at the executive level rather than at the task level, continually modifying the schema hierarchy. Task-level actions are executed by the motor parts of the schemas.

The behavior generation approach we favor is not a traditional planning approach but rather one of "adaptive dynamics" in which the system tries to use one or more simple behaviors to generate a complex behavior (Benjamin, 2000; Lyons & Hendriks, 1995a). For example, if the robot is trying to pick up a ball that is rolling on the floor, it activates three schemas: the grasping schema, the move-arm schema and the tracking schema. The grasping and move-arm schemas are connected to the tracking schema. As the rolling ball is tracked by the tracking schema, information is fed by the tracking schema to the two motor schemas to control the rate of grasping and movement, so that the ball will be caught.

This approach relies very heavily on the representation used by the system. ADAPT's approach to analysis and synthesis is to reformulate its knowledge to try to find simple schemas that generate complex behaviors (Benjamin, 1992; Benjamin, 1994). One of the central research goals of this project is to evalute methods for solving problems by reformulation, as explained by Duncker (1945) and illustrated by Amarel (1968).

Decomposition and Reformulation

The complexity of robot perception and action arises in part from the need for a robot to perceive and act simultaneously on several space and time scales, ranging from small scales of millimeters and milliseconds to large scales of kilometers and hours. Each of these scales seems to possess a complex structure, and each can potentially constrain any other. Yet a robot must perceive and act in real time.

Our decomposition algorithm finds for each task a hierarchy of local neighborhoods, and reformulates the task in terms of local coordinate systems for these neighborhoods. The relevant properties for system formulation are those properties that hold locally everywhere within the neighborhoods. Simple programs can measure these properties and maneuver within the neighborhoods. Under certain conditions, the neighborhood structure becomes a substitution tiling (Senechal, 1995), and its self-similarity permits the simple programs to be effective at several scales, creating a hierarchical structure that is simple and local at each scale, but capable of generating very complex global behaviors. Fortunately, substitution tilings appear to be ubiquitous in the physical world (Senechal, 1995).

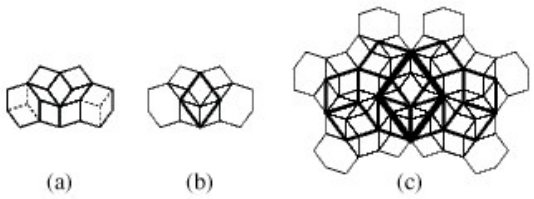
Local coordinate systems define the perceptual spaces for the robot. The axes in each coordinate system correspond to a set of features for describing the state space. Associated with the local symmetries of the local coordinate system are invariant subspaces of states (states on which particular combinations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

Thus, this approach integrates perception, action and communication in a principled way by connecting their local symmetries and invariants. Our previous work into this structure has shown that it exists in motion, perception, communication and planning (Benjamin, 1992, 1994, 2000). Barnsley (1996) has shown that this self-similar structure exists in images and has developed fractal compression methods that are commercially deployed in the areas of image processing and communications.

Natural language is another example of the interaction of top-down processing (overall discourse meaning, major syntactic and semantic structures) and bottom-up processing (word meaning, speech recognition). The use of local symmetries in the analysis of natural language is well understood (Eilenberg, 1976) (Lallement, 1979) (Petrich, 1984).

Constraint satisfaction problems can also exhibit this self-similar structure (Benjamin, 1997). A simple illustrative example is the tiling of the plane using Penrose tiles. The Penrose tiling problem is to cover the two-dimensional plane (or a portion of it) with no holes using copies of two rhombus-shaped tiles, one of which is thinner than the other. In (a) below we see five copies of the fatter tile and two copies of the thinner tile (ignoring the dotted shapes).

This is a typical combinatorial search task, and the solutions of the Penrose tiling placement problem are typically found by depth-first search with considerable backtracking (when holes are created that cannot be filled), but Steinhardt (1996) has shown that nine copies of the local pattern shown at left in (a) below can be replicated and overlapped to give the pattern in (c) at right. This larger pattern is *self-similar* to the original pattern using the substitution given in (b). In this substitution, the group of tiles is viewed as one fat rhombus at a larger scale. Copies



of this larger rhombus can them be composed as in (a) to make even larger rhombuses, etc. Steinhardt shows that all

Penrose tilings arise in this way, and thus Penrose tilings are locally everywhere isomorphic to the pattern in (a), at many scales.

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

This self-similar structure maps perfectly onto the universal subgoaling of Soar, and is the main reason that Soar was chosen as the basic component of ADAPT.

Search Control in ADAPT

ADAPT possesses a very general mechanism for selecting one action to perform. This mechanism is essentially the same as Soar's subgoaling mechanism, because the emphasis in ADAPT is on problem solving by search, as it is in Soar. The philosophy behind this mechanism is that the system should be capable of bringing all its knowledge to bear on the central issue of selecting its actions. ACT-R does not permit this, instead using a Bayesian mechanism to select actions based on their anticipated gain.

However, much robotics research uses such nonsymbolic mechanisms. In particular, there is a very large body of robotics research using neural networks, so a general robotic cognitive architecture should be able to model nonsymbolic reasoning and learning mechanisms, which Soar does not. Although the mechanism ADAPT uses to select actions is Soar-style subgoaling, the presence of the declarative schema memory opens the possibility for ADAPT to implement ACT-R nonsymbolic mechanisms, too. We plan to accomplish this by storing the information that ACT-R uses to compute utilities with the declarative schemas, so that each schema has stored with it the estimate of the probability that it will lead to eventual achievement of the current goal and the estimate of the cost to achieve the goal. Both probability and cost will be measured in time, as in ACT-R. The utility computation is performed explicitly with ADAPT actions, rather than in the mechanism as ACT-R does.

When selecting among multiple schemas, ADAPT reaches an "impasse" just as Soar does and subgoals to choose one schema. It can choose either to use the Bayesian estimate or to search. This decision can be made based on factors such as the time available. This generalization of the selection mechanisms of ACT-R and Soar permits ADAPT to model flexible selection strategies that combine symbolic and nonsymbolic processing.

For example, in the example given above of the rolling ball, the robot might need to decide whether to take a step closer to the path of the rolling ball to make it easier to pick up. If this decision must be made immediately, the robot will use its estimated gain for such movements in general. If there is time to deliberate, the robot can reason about the length of its arm and how far it is from the path of the ball to decide if it will have a good chance to succeed.

Learning in ADAPT

There are two different methods of learning in ADAPT: procedural learning of search control and inductive inference of schemas.

ADAPT utilizes a method of learning search control that is borrowed from Soar. ADAPT generates procedural "chunks" when goals are satisfied in exactly the same way that Soar does. These chunks are productions whose left-hand sides contain all the working memory elements that were referenced in making the search-control decision, and whose right-hand side is the decision. It is well understood how Soar speeds up its search control by learning search control chunks (Rosenbloom, Laird & Newell, 1993).

A search-control chunk that ADAPT learns may use the Bayesian estimate to make the choice of action, in which case the chunk performs in one step the same choice that ACT-R would make. Or the chunk may compile the results of a search of alternatives, in which case the chunk performs just as a Soar chunk does.

ADAPT can learn a chunk that combines these two methods to select a schema based on a combination of gain and search.

The declarative representation of schemas permits ADAPT to transform schemas in many ways. In particular, ADAPT can perform inductive inference on schemas, e.g. by replacing a constant with a variable or by enlarging an interval of permitted numeric values. These changes are performed by operators that examine the execution history and hypothesize more general schemas that are added to the declarative memory. As these new schemas are tried in future situations, their successes and failures are recorded in the schemas.

An Implementation of ADAPT

A simple, initial implementation of ADAPT has been completed. Testing has just begun using a Pioneer P2 robot that is equipped with stereo color vision, microphone and speakers, sonars and touch sensors.

The implementation is within the Soar system, because of the similarity between ADAPT and Soar. A declarative memory has been added to Soar for the general schemas, together with a set of operators that instantiate a general schema and transform the instance into a set of productions. Additional operators have been added to start, pause and stop schemas. Schemas are executed by a runtime system that implements the RS schema system (Lyons & Arbib, 1989) in the Colbert language, which is a behavior-based language created by Kurt Konolige and provided with Pioneer robots.

Currently, ADAPT has access only to sonar and bump-sensor readings. This permits simple navigation and

pushing behaviors, but nothing more. Our plans are to complete the integration of language and vision capabilities within two months. The existing version of ADAPT has a cycle time of 50ms and is very successful at guiding the robot at basic navigation tasks such as moving from one room to another and avoiding obstacles.

The language component will be provided by NL-Soar (Lonsdale, 1997, 2001; Lonsdale & Rytting, 2001a, 2001b), which is currently being ported to Soar8 from Soar7 as part of our collaboration with Brigham Young University. It will be available in May and will be integrated into the system.

The vision component consists of two pieces: a bottom-up component that is always on and is goal-independent, and a top-down active component. Both components exist but the full integration within ADAPT is not yet complete. Our target for a full integration is March, 2005.

Testing and Evaluating ADAPT

We have selected an important and 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 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, especially those that have been examined by cognitive psychology. For example, we can create an isomorph of the Towers of Hanoi task by having three narrow enclosures and adding the constraint that no object can be in front of a shorter object (so that all objects are always visible by the observer). (see Figure 1) The three enclosures act as the three pegs in the Towers of Hanoi, and the constraint is isomorphic to the constraint that no disk can be on a smaller disk in the Towers of Hanoi. 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. The front area must be kept clear so the robot can move; there can be no boxes left in this area.

This creates a situation in which the robot can be presented with a Towers of Hanoi problem in a real setting with perceptual and navigational difficulties, rather than just as an abstract task. This permits us to evaluate the robot's problem-solving and learning capabilities in a way that permits comparison with human data.

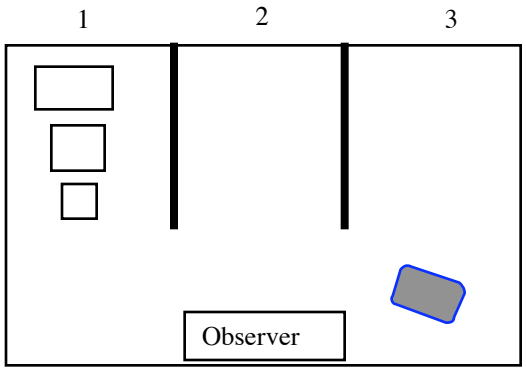


Figure 1

Similarly, we can embed bin-packing problems by making them enclosure-packing problems. Also, we can embed sorting problems and block-stacking problems. In this way, many existing puzzles and tasks can be embedded in a realistic setting and posed to the robot.

Summary

The fields of cognitive psychology and robotics have much to offer each other. The development of robot cognitive architectures is an attempt to apply the results of cognitive modeling to the difficult problems faced by robotics research. ADAPT is a cognitive architecture specifically designed to permit robotics researchers to utilize well-known robotics research in areas such as active perception and adaptive dynamics within a cognitive framework. This architecture is still in its infancy, and in particular has not yet been integrated with vision or language. The goals of this research are to expand the capabilities of robots and simultaneously to expand and generalize the capabilities of existing cognitive models.

References

Allen, J.F., 1981. "An interval-based representation of temporal knowledge", in Hayes, P. J., editor, IJCAI, pp. 221-226, Los Altos, CA.

Amarel, Saul, 1968. On Representations of Problems of Reasoning about Actions, in Michie (ed.) Machine Intelligence, chapter 10, pp. 131-171, Edinburgh University Press.

Anderson, J. R., 1996. ACT: A simple theory of complex cognition. American Psychologist , 51, 355-365.

Arbib, M. A., 1993. Allen Newell, Unified Theories of Cognition. Artif. Intell. 59(1-2): 265-283.

Arbib, M. A., 1992. Schema Theory, in S. C. Shapiro (Ed.), Encyclopedia of Artificial Intelligence (2nd edition), Wiley-Interscience.

Arkin, R., 1998. Behavior-Based Robotics, MIT Press, Cambridge, MA.

Barnsley, Michael F., 1996. "Fractal Image Compression", Notices of the American Mathematical Society, Volume 43, Number 6, pp. 657-662, June.

Benjamin, D. Paul, 2000. "On the Emergence of Intelligent Global Behaviors from Simple Local Actions", Journal of Systems Science, special issue: Emergent Properties of Complex Systems, Vol. 31, No. 7, 861-872.

Benjamin, D. Paul, 1997. "A Decomposition Approach to Solving Distributed Constraint Satisfaction Problems", Proceedings of the IEEE Seventh Annual Dual-use Technologies & Applications Conference, IEEE Computer Society Press.

Benjamin, D. Paul, 1994. "Formulating Patterns in Problem Solving", Annals of Mathematics and AI, 10, pp.1-23.

Benjamin, D. Paul, 1992. "Reformulating Path Planning Problems by Task-preserving Abstraction", Journal of Robotics and Autonomous Systems, 9, pp.1-9.

A. Blake and A. Yuille, eds, 1992. Active Vision , MIT Press, Cambridge, MA.

Brooks, R. A. 1986. "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, Vol. 2, No. 1, pp. 14–23, March.

Eilenberg, Samuel, 1976. Automata, Languages, and Machines, Volumes A and B, Academic Press.

Duncker, Karl, 1945. On Problem Solving, in Dashiell, John, (Ed.) Psychological Monographs, Greenwood Press, Westport, CT.

Garcia-Martinez, R., and Borrajo, D., 2000, "An Integrated Approach of Learning, Planning, and Execution," Journal of Intelligent and Robotic Systems 29(1): 47-78.

Gopher, D., & Donchin, E., 1986. Workload: An Examination of the Concept, in K.R.Boff, L. Kaufman, & J.P.Thomas (Eds.), Handbook of Perception and Human Performance, Vol. II: Cognitive Processes and Human Performance, pp. 41.1-41.49, New York, Wiley.

Kieras, D.E., Wood, S.D., and Meyer, D.E., 1997. Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task, ACM Transactions on Computer-Human Interaction 4, 230-275.

Laird, J.E., Newell, A. and Rosenbloom, P.S., 1987. "Soar: An Architecture for General Intelligence", Artificial Intelligence 33, pp.1-64.

Lallement, Gerard, 1979. Semigroups and Combinatorial Applications, Wiley & Sons.

D.Long and M.Fox and M.Hamdi, 2002. "Reformulation in Planning," SARA 2002 (Springer Verlag LNCS series volume).

Lonsdale, Deryle, 2001. "An Operator-based Integration of Comprehension and Production," LACUS Forum XXVII, pages 123–132, Linguistic Association of Canada and the United States.

Lonsdale, Deryle and C. Anton Rytting, 2001a. "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.

Lonsdale and C. Anton Rytting, 2001b. "Integrating WordNet with NL-Soar," WordNet and other lexical resources: Applications, extensions, and customizations; Proceedings of NAACL-2001; Association for Computational Linguistics.

Deryle Lonsdale, 1997. "Modeling cognition in SI: Methodological issues. International Journal of Research and Practice in Interpreting", 2(1/2): 91–117.

Lyons, D.M. and Hendriks, A., 1995a. "Planning as Incremental Adaptation of a Reactive System", Journal of Robotics & Autonomous Systems 14, pp.255-288.

Lyons, D.M. and Hendriks, A., 1995b. "Exploiting Patterns of Interaction to Select Reactions", Special Issue on Computational Theories of Interaction, Artificial Intelligence 73, 1995, pp.117-148.

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

Lyons, D.M. and Arbib, M.A., 1989. "A Formal Model of Computation for Sensory-based Robotics", IEEE Transactions on Robotics and Automation 5(3), June.

D.M. Lyons, and A.J. Hendriks, 1991. Reactive Planning. Encyclopedia of Artificial Intelligence, 2nd Edition, Wiley & Sons, December.

Newell, Allen, 1990. Unified Theories of Cognition, Harvard University Press, Cambridge, Massachusetts.

M. Nicolescu and M. Mataric, 2000. "Extending Behavior-based System Capabilities Using an Abstract Behavior Representation", Working Notes of the AAAI Fall Symposium on Parallel Cognition, pages 27-34, North Falmouth, MA, November 3-5.

Petrich, Mario, 1984. Inverse Semigroups, John Wiley & Sons, Inc., New York.

P. S. Rosenbloom, J. E. Laird, and A. Newell, (1993). The SOAR Papers. MIT Press.

M. Senechal, 1995. Quasicrystals and Geometry, Cambridge University Press.

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

Steinhardt, Paul J., 1996. New Perspectives on Forbidden Symmetries, Quasicrystals, and Penrose Tilings, Proceedings of the National Academy of Science USA, Vol. 93, pp. 14267-14270, December.

Manuela M. Veloso, Jaime Carbonell, M. Alicia Perez, Daniel Borrajo, Eugene Fink, and Jim Blythe, 1995. "Integrating planning and learning: The Prodigy architecture," Journal of Experimental and Theoretical Artificial Intelligence, 7(1):81--120.