Lessons Learned from the Animate Agent Project (so far)

R. James Firby *
Artificial Intelligence Laboratory
Computer Science Department
University of Chicago
1100 East 58th Street
Chicago, IL 60637
firby@cs.uchicago.edu

Abstract

This paper gives a brief overview of the Animate Agent Project at the University of Chicago, describing skills, RAPs, and a recently added spatial planning module. The RAP system takes in symbolic goals and refines them into sequences of continuous skills that move the robot in real-time. The RAP system uses the spatial module as an assistant when spatial problems arise during goal refinement. As an example, the skills, RAPs, and spatial planning actions for a simple trash cleanup task are also briefly described.

The paper finishes with answers to the seven symposium questions.

The Animate Agent Project

The Animate Agent Project at the University of Chicago is aimed at building robotic and software systems for carrying out a variety of everyday goals in realistic, everyday environments (Firby & Swain 1991). A major goal of the project is to create a software architecture that allows existing plans and behaviors to be used as building blocks for additional tasks.

The architecture currently being used consists of three major components: the RAP reactive plan execution system, a modular, skill-based control system, and a spatial planning subsystem incorporating a simple path planner. Both the control system and the spatial planning system are essentially under the control of the RAP system.

Skills

The skill-based control system connects the robot to the world. It consists of a collection of action routines for moving the various robot actuators and a collection of sensing routines (primarily visual) for finding and tracking objects in the world. The action routines are designed to take information from the sensing routines in real-time and the robot takes action by enabling a set of action and sensing routines that work together in coherent feedback loops.

For example, the robot has a routine for moving in a particular direction and it has a variety of visual routines for tracking targets and giving their direction. Combining these two, the robot can be made to approach and stop near a target like a desk or trash can or it can be made to follow a moving target like a person and so on. Careful selection of the active action and sensing routines can be used to cause the robot to carry out a variety of task steps (Firby 1992).

RAPs

The RAP reactive execution system manages the skill-based control system by enabling, disabling, and configuring the available routines. It also interprets signals from the control system as the end of a given task step, a problem, or a new piece of information about the world. The RAP system takes in high-level symbolic goals, such as "pick up all the trash in this room", "deliver this paper to Bob", or "follow me" and refines them into a sequence of appropriate skill sets. This refinement uses a hierarchical library of known methods for arrying out subgoals in particular situations. The RAP system offers a language for representing symbolic, plan-like knowledge for carrying out tasks in the world and a mechanism for refining those plans at runtime while dealing with failures and untoward events (Firby 1994).

Spatial Planning

The spatial planning module is a special purpose problem solver under control of the RAP system. The RAP system is very good for reactively refining plans and for dealing with contingencies. However, it is not very good for reasoning about spatial relationships such as deciding which direction to go around an obstacle or keeping track of the floor area already cleaned up. The spatial planning module is designed to answer these questions for the RAP system. The module defines

*This work was supported in part by ONR contract N00014-93-1-0332 and ARPA contract N00014-93-1-1394.

a limited set of operations and the RAP system uses those operations in its plans. The method of interaction is similar in idea to that of Ullman's visual routine processor (Ullman 1984). In effect, the module acts like a "spatial calculator" that the RAP system can use to help it carry out tasks with spatial components (like clearing all of the trash off the floor, or traveling to Bob's desk).

Problem Solving in Context

Using this architecture, the RAP system uses the control system to make things happen in the world, and the spatial module to work out the spatial relationships among objects. We anticipate using a variety of other special purpose problem solvers to work out other problems. Putting the reactive RAP system in control of these modules allows them to deal with just the problems they are good at and change problem focus in response to changing situations. Gat suggested a similar arrangement in the ATLANTIS architecture (Gat 1992).

A Trash Cleaning Example

As an example of the Animate Agent Architecture at work, consider the problem of cleaning up all the trash on the floor. This task is currently being investigated extensively at the University of Chicago. The RAP task descriptions in the system describe cleaning plans at various levels of abstraction as well as methods for moving the robot from one place to another, finding and tracking an object, picking an object up, and putting an object in the trash. The skill system contains the building blocks for actually moving the robot actuators, including skills for moving in a given direction while avoiding obstacles, turning to face a particular direction, finding an object visually, tracking an object, and reaching toward an object. The spatial planning module keeps track of the floor area cleaned so far, and the locations of pieces of trash that have been seen but not yet dealt with.

Skills for Picking Up Trash

In particular, the robot's skill system includes routines for:

- moving to an (x,y) position while avoiding obstacles
- panning to look at an (x,y) position
- turning to face an (x,y) position
- lining up with an (x,y) position (to prepare for a grasp)
- moving the arm up and down
- opening and closing the gripper
- tilting the pan/tilt head to given angle
- finding and identifying an object
- tracking an object, generating estimated (x,y) position
- tracking a position based on dead-reckoning, generating estimated (x,y)

These skills can be combined to perform simple actions like: find a piece of trash, track a piece of trash and move towards it, align with a piece of trash, and pick a piece of trash up. By tracking the trash while approaching and aligning with it, the system can compensate for errors in the robot's motion and errors in initial estimates of the trash item's location.

The only way that the robot can act is by enabling some combination of these routines. While a combination of routines is enabled, it has complete control over the behavior of the robot.

RAPs for Picking Up Trash

The RAP methods used by the system consist of plans for choosing and enabling sets of skills. In particular, there are RAP plans for:

- moving to a location
- turning to face in a direction
- searching for an object within visual range
- moving to an object while tracking it
- aligning with an object
- moving forward and backward
- picking an object up
- dropping an object

These plans build on one another into plans for:

- finding and picking up a nearby piece of trash
- finding a trash can
- going to a trash can and dropping in a piece of trash
- finding a nearby piece of trash and putting it in a trash can
- searching new areas of the room for trash

Finally, these subplans are built into high-level RAP methods for cleaning up all the trash in a room.

These plans are coded hierarchically so the lower-level plans can be used for a variety of high-level goals. For example, we are also working on the task of going to a person's desk (at their request), taking a can from the person, and dropping it into the recycling bin. The high-level RAP methods for this task can be constructed almost entirely from the lower-level plans already used for cleaning up the room.

Spatial Planning for Trash

The spatial module used by the RAP system to help with these plans includes the following operations:

- entering new room
- add item to current room memory
- remove item from current room memory
- next navigation action to get to item (or location)
- mark room area
- clear room area
- location of nearest/biggest marked/unmarked area

Currently, the spatial module divides the world up into rooms and the has a different x,y coordinate system for each room. The RAP system informs the module each time it moves into a new room and the module returns an appropriate update to the robot's dead-reckoning navigation coordinates. The RAP system also adds and removes items from the current room memory as it picks them up and puts them down. It can also ask the module how to get from the current location to any item or location the module knows about. The module builds a plan to get to the appropriate room using a graph search and a more detailed route plan in the current room using a simple configuration space planner. The module also allows the RAP system to mark and clear various areas in the current room so the RAP system can keep track of those areas it has searched for trash and those it has not. None of these operations are specific to cleaning up trash and they form the basis of a variety of navigation-based plans.

All of the skills mentioned above are currently implemented on the Chip robot at the University of Chicago. The RAPs for cleaning up nearby trash are also implemented (and actually work, with the robot having now picked up hundreds of pieces of trash). The spatial module is still in its infancy and has not yet been fully integrated with the RAP system so there are currently no high-level RAPs for picking up all of the trash in a room. Currently, when the robot cannot see a new piece of trash immediately after dropping something in the trash can, it does not have a good idea of where to look next.

Symposium Question Answers and Comments

Let me try and address the questions raised in the call for papers within the Animate Agent Architecture and the trash cleaning example outlined above.

1) Is there a need for central behavior coordination?

Yes, there is a need for central behavior coordination but not at the level this often means. Looking at the example skills mentioned above, it is clear that a number of them are incompatible and should not be enabled at the same time. In the animate agent system, it is up to the RAP system to make sure it doesn't enable conflicting skills at the same time. There is no provision for the usual notion of very low-level behavior pieces that routinely give conflicting actuator commands that need arbitration. Correct coordination at that level is impossible if the system has no knowledge of higher-level goals. Since the RAP system has this knowledge, and we don't want to duplicate it in an arbitration network somewhere, we rely on the RAP system to coordinate things properly.

Using the RAP system for behavior coordination does place more of a burden on the skill writer. Care must be taken to ensure that skills are designed to be used in a modular way and that when skills run together they do not interfere with one another. However, this care is required anyway either in writing the skills in the first place, or in crafting the arbitration network afterward.

2) Does an agent need to know natural language before it can make use of human expertise at run time?

This is an interesting question and I'll basically sidestep it. There are several different reasons that an agent might need to interact with human beings. First, the agent may simply be in an environment with people and have to cope with what they do. To a large extent this problem can be handled by simply treating people as moving obstacles and their actions as exogenous changes in the world. Of course, smoother interaction can take place with better communication but it isn't strictly necessary.

Second, the robot may be designed to accept new goals from people out in the world. The obvious way to do this without natural language is via the person knowing the goals the robot can achieve and selecting one of those, either by typing in a new RAP to execute or selecting something from a menu. These are fairly unsatisfactory because they require sophisticated users and some sort of I/O device other than voice. Often, however, the I/O device constraint can be dropped as long as the user remains relatively sophisticated. There are a number of speech understanding boards available that allow a user to talk to the robot (we don't have one yet). If the user can remember a simple speech-based code, like a restricted vocabulary, it is relatively easy to map user utterances to RAP-level goals.

Third, a robot may need to coordinate its activities with a human being. Natural language is good for

this but even people often use gestures and grunts and signals that aren't really language. We are looking at some simple gesture recognition to coordinate following strategies with a human but we have only just begun to implement them.

Finally, a human may want to communicate a new plan to the robot. Unless the human wants to try and show the robot the entire plan, there are only two ways to communicate plans: via programming, or via natural language. One of the purposes of natural language is to allow agents to exchange plans without knowing each other's internal representation and programming language. Thus, if we want robot's to take advantage of human planning expertise without having to program it in, the robot will have to know how to understand natural language.

Natural language understanding will be extremely useful for a robot, particularly in negotiating goals and plans with humans, but a robot can manage a lot of human interaction without it. I believe the primary constraint that natural language places on robotic research right now is in the area of knowledge representation. Our robots will never be able to converse with us in a meaningful way if they do not perceive and represent the world the way people do (at least at some level). For example, if the robot does not represent the notion of an object, language will be very hard to deal with because it is extremely object oriented.

3) How much internal representation of knowledge and skills is needed? Is more than one representation formalism needed?

Currently, the Animate Agent Architecture has very little provision for planning and learning so there is little need for the system to represent detailed knowledge about its own skills. However, we are beginning a project aimed at learning the behavior of new objects. This will require learning some representation of the effects of various actions on the object involved. RAPs will then have to make future action choices that take this new knowledge into account. Without some representation of the effects of actions on objects, a robot will be unable to learn how to deal with new objects.

Learning the effects of actions on objects is only one example of representing knowledge about one kind of thing (what skills do when applied to objects) so that another kind of thing can use the info (RAPs can choose appropriate actions in the future). The current system uses three types of representation: skills for capturing task control knowledge, RAPs for capturing symbolic execution methods, and the spatial module for reasoning about spatial relationships. Each of these representations is designed to make certain operations easier: skills allow feedback loops to be represented and assembled, RAPs allow symbolic plans to be represented and executed, and the spatial module represents simple spatial relationships and solves specific simple spatial problems.

Multiple representations allow each type of knowledge to be represented in a clear, easy to use form. The problem with multiple representations is figuring out the interface between them. In particular, we don't want to represent all knowledge in every representation scheme. The Animate Agent Architecture makes a strong claim about the interface between skills and RAPs (enabling and disabling only) but as yet has little to say about the interface between RAPs and special problem solving modules.

4) How should the computational capabilities of an agent be divided, structured and interconnected? How much does each level/component need to know about the others?

The basic Animate Agent Architecture defines three components: the skills system, the RAP system, and special problem solving modules. The split between the skill and RAP system has several justifications. First, skills are described much better as continuous processes and the Animate Agent Architecture encourages them to be coded that way. Second, any architecture needs an interface between those processes best described as continuous, such as following a person, and those best described as symbolic, such as a plan for finding trash, picking it up, and putting it in the trash can. The interface between the RAP and skills systems consists of enabling an appropriate set of skills and then waiting for an event in the world before proceeding to the next set. This is a natural way to partition continuous processes into sets that can be treated discretely. Third, skills can easily be distributed across multiple machines because the interface between skills and the RAP system is simple and rigidly defined. Chip runs skills on three different computers and the RAP system on a fourth.

The RAP system "needs to know about" the skill system in some sense, just as it "needs to know about" the special problem solving modules to which it has access. Currently, this knowledge resides in the methods used by the RAP system to get things done, or more precisely, in the heads of the programmers who write those methods. As mentioned above there is a project beginning to look at learning how objects act in the world and that project is changing the way the RAP system encodes methods so that the connection between actions and their effects on objects is made more evident. Another project (joint with Charles Martin) is looking at combining RAP method representation with a natural language understanding system. This

work also changes the way RAP methods are encoded so that their connection with object effects is more direct. I believe that the primary representation of action in the system (in this case RAPs) must have a clear representation of how its various "symbolic" actions affect objects in the world. In the implemented system this is not the case.

5) What types of performance goals and metrics can realistically be used for agents operating in dynamic, uncertain, even actively hostile environments?

This question is one that plagues most attempts to get robust behavior out of a system in a dynamic, uncertain world. To a large extent the real measure of success is how long the system can run and successfully achieve goals before it crashes in some way. Failing to achieve any particular goal is not a good measure because systems dealing with the real complexities of the world will always be heuristic and make mistakes. The trick is to try and build a system that can recover from these mistakes and continue on with future goals. The system only really fails when it needs to be restarted, or it gets so badly confused it cannot achieve new goals in a reasonable manner.

Unfortunately, this measure of success is terribly vague. I hope that the seminar will manage to suggest a quantitative measure of success beyond a subjective measure of generality and robustness.

6) Why should we build agents that mimic anthopomorphic functionalities?

Experience with various projects, including the Animate Agent project, suggests to me there are only two compelling reasons to build agents that mimic anthopomorphic functionalities to:

- Facilitate man/machine communication.
- Be able to cope with man-made environments.

As mentioned above, if robots are to use natural language, they must represent the world in ways that support language use. I believe that this means robots must perceive and represent the world in terms of objects since language relies very heavily on this concept. However, this does not require that robots actually take action in anthropomorphic ways. My experience is that people have very little intuition about how they carry out tasks below the symbolic planning level so there is little advantage in trying to build robots that take action like people. It is much more important that they *think* like people so that natural language makes sense. (I confess I don't really know what this means).

A much lesser reason for building anthopomorphic robots is so they can use tools and machines designed for people. Robots that can't climb stairs are hard to use in many situations (although recent legislation for the disabled is helping considerably in public buildings), as are robots that can't push buttons, or pick up ordinary things. This reason has little theoretical import, however.

7) How can a given architecture support learning?

This question is not addressed well by the Animate Agent Architecture since it currently does not do any learning. However, learning is a crucially important aspect of architecture design. If an architecture does not allow at least the idea of learning it is obviously useless in the long run. In the past, I have made the argument that the representation used by the RAP system allows new methods to be added and used, although I've never suggested a mechanism for doing so. I hope that will change soon. On the other hand, there is currently no real provision for learning within the skill system. We are looking at various adaptive control algorithms to reduce or eliminate calibration issues in the skills we have, but there is no real way of building new skills automatically. Similarly, problem solving modules are still at the black art stage.

I believe learning is a central issue and one any architecture must pay attention to. However, I have little to offer in the way of learning systems at this time.

References

Firby, R. J., and Swain, M. J. 1991. Flexible task-specific control using active vision. In *Sensor Fusion IV: Control Paradigms and Data Structures.* Boston, MA: SPIE.

Firby, R. J. 1992. Building symbolic primitives with continuous control routines. In *First International Conference on AI Planning Systems.*

Firby, R. J. 1994. Task networks for controlling continuous processes. In *Second International Conference on AI Planning Systems.*

Gat, E. 1992. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Tenth National Conference on Artificial Intelligence.* San Jose, CA: AAAI.

Ullman, S. 1984. Visual routines. *Cognition* 18.