Life after planning and reaction

Ian Horswill and Lynn Andrea Stein
Artificial Intelligence Laboratory, Room 733
Massachusetts Institute of Technology
Cambridge, MA 02139
{ian||las}@ai.mit.edu

Abstract

Recent action selection architectures rely on planning, reaction, or both, ignoring the much wider spectrum actually available. This paper recasts the action selection problem in terms of projection, commitment, and abstraction.

1 Introduction

The problem of *action selection*, selecting appropriate actions to perform in a given situation, has received a great deal of attention in AI. Much of the recent discussion about action selection has concerned the distinction between "planning" and "reaction," terms with which few researchers have been entirely happy. While there is broad consensus that planning and reaction are two points on a spectrum rather than opposing sides of a dichotomy, we believe that the use of these terms has encouraged the conflation of other, more basic, distinctions. Furthermore, it has encouraged researchers seeking intermediate points in the planning/reaction spectrum to consider mostly hybrid systems formed by fusing pure planning with pure reaction, rather than searching for truly novel algorithms.

This paper is a purely personal view of the field as we see it. Based on our own experiences in designing and building mobile robots, we believe that there are a wide variety of useful and as-yet-unexplored design possibilities. We hope that the following discussion will encourage others to consider non-traditional as well as traditional approaches to designing the control of robotic systems.

2 Projection and Commitment

Rather than a spectrum from planning to reaction, we believe that action selection systems are better characterized in terms of the orthogonal axes of *projection* and *commitment*.

We will use *projection* to mean the use of knowledge about the expected outcomes of hypothetical actions for choosing actual actions. In traditional planning, projection is explicit. It typically involves the "simulation" of various courses of action, though this simulation may be at an abstract level and is invariably reversible in a way that genuine action is not. For example, STRIPS operators are not real actions, but model certain properties of those actions for the purposes of searching the space of possible courses of actions.

We will say that an agent's *commitment* to an action or series of actions is the space of situations in which the agent will always take the action. For example, a traditional planner will commit to a sequence of actions unless an execution monitor indicates a need for replanning. A conditional planner commits to a particular action (or sequence of actions) only in case certain conditions are met at the time that the action is to be taken. A universal plan is a fully conditionalized plan in which every step of every action sequence depends on conditions at that time. All three of these "planners" use projection to select an appropriate action sequence, though in the case of the universal plan, the projection is done off-line.

The advantage of projection is—under most circumstances—increased knowledge; the cost incurred is the time and energy devoted to projecting rather than acting. The benefit of remaining uncommitted is increased flexibility to respond to potentially unanticipated situations; its cost comes when a new course of action must be decided upon.

3 Projection and commitment are orthogonal

People sometimes assume that projection requires commitment. This is easy to do, since much of the literature divides action selection algorithms firmly into the categories of "reactive" systems (no projection, no commitment), and "planning" systems (high projection, high commitment). However, the two axes really are orthogonal. Let us, for rhetorical purposes, define a reactive system to be any system whose control structure has the following form:

```
do forever
  action = reactively-choose-action(current-situation)
  perform action
end
```

and define a planner/executive system to be any system whose control structure has this form:

```
do forever
  g = choose-goal()
  action-sequence = plan-action-sequence(goal,current-situation)
  for each action in action-sequence do
    perform action
  end
end
```

The planning system outlined above probably performs a great deal of projection to construct `action-sequence`. It also maintains a great deal of commitment—once `action-sequence` has been selected, its execution continues (roughly) uninterrupted. However, we can trivially modify the planner to give it the form of a reactive system, simply by defining `reactively-choose-action()` to be:

```
    first-action(plan-action-sequence(goal,current-situation))
```

so that the agent control structure looks like:

```
do forever
  action = first-action(plan-action-sequence(goal,current-situation))
  perform action
end
```

Now we have a system which on every "clock tick" considers the current situation, projects an action sequence which will accomplish the desired goal, executes the first action of that sequence, and throws the rest of the sequence away. On the next tick, it recomputes the sequence from scratch.[1] This system performs as much projection as the planner—or more, since it is re-projecting at each time step—but makes minimal commitment to the projected action sequence—making it responsive to environmental change, but at the cost of high projection overhead.

The original planner's projection is implicit in the fact that `plan-action-sequence` examines long sequences of actions rather than just isolated actions. Its commitment comes from a *separate* source, the fact that it stores *and executes* these complete action sequences before reevaluating the situation. Projection and commitment are correlated in traditional systems not for logical reasons, but for engineering reasons: when projection is expensive, programmers "cache" the action sequence and only recompute it when absolutely necessary.

We can consider other points in projection/commitment space. Projection without commitment is often a useful strategy in domains where additional knowledge is likely to be available later. The "reactivized" planning system discussed above is an extreme example of projection without commitment. Many game playing programs project (e.g., through minimaxing) to choose a next move, but do not commit to any sequence of later moves. Indeed, such commitment would be foolish, as the opponent's choice of move may make a completely different sequence more appropriate. Conditional planners similarly exploit knowledge not available at planning time to permit later tailoring of action selection, though in general conditional planners involve at least a moderate degree of commitment.

Commitment without projection involves selecting a sequence of actions without first analyzing its potential consequences; while often a bad idea, it is far from unimaginable. Indeed, in biological systems, such commitment with-

[1] Of course, the clock ticks might be measured in hours...

out projection is common and know as a fixed action pattern [5][16].[2] Even within robotics, certain control—especially of industrial assembly, etc.—may have a similar form.

4 Abstraction and projection/commitment are orthogonal

It is also easy to assume that projection and commitment are required for "high level", "abstract" operations, while reactivity is required for "low level", high speed operations. Such a division of labor is attractive for a number of reasons:

- Level of abstraction is correlated with time scale. The low levels often need to run fast, so reactive systems are usually chosen over deliberative planners.
- Low level actions are often unpredictable: getting from one end to a corridor to the other can be implemented fairly reliably, allowing us to abstract it as a single operation, but the instantaneous responses of motors to movement commands may be much less predictable, particularly over hard-to-model terrains such as carpets. Again, reactive systems are typically chosen to cope with this unpredictability.
- Abstract actions are, of necessity, formed out of chains of concrete actions. Forming chains of actions is precisely what planners are good at.

A common design in current autonomous systems is the so-called "three level architecture" or "TLA" in which which different levels of abstraction are rigidly divided up amongst different types of implementation machinery:

- low level motor control is performed by continuous servo loops,
- medium level control is performed by discrete-symbol reactive systems, and

[2] For example, upon finding a misplaced egg outside its nest, the greylag goose will roll the egg back into the nest by repeatedly making pushing motions with its beak. This fixed action pattern—the beak pushing—continues even if the egg is removed [16].

- high level control is performed by a conventional planner.

There are of course many variations. For examples, see [7][4][12].

4.1 Mobile robot hallway navigation: (projection, followed by reaction)

For example, the mobile robot community (the authors included) have been increasingly wed to hierarchical models of navigation in which low level navigation is performed by reactive path following and collision avoidance, while high level, goal directed navigation is performed by search of a topological representation of space (see e.g. [12][15]). In short, high level projection, followed by low level reaction. In most familiar environments, this is a pretty good architecture. However, it is not difficult to construct other environments which call for qualitatively different methods. They need not even be particularly pathological.

4.2 Navigating grids: (reaction, reaction)

For example, consider driving through midtown Manhattan with a moment-to-moment traffic report playing on the radio. Here, the topology of the world is extremely simple, but the dynamics can be rather intense. Clearly, one needs do be able to do reactive path following and collision avoidance at the low level. But what about the "high level" choice of path sequences? Projection buys you relatively little in terms of path optimality: if you're going the wrong way, you can can almost always take the next turn (or the second, if the next is one way the wrong way). Also, large scale traffic patterns change from moment to moment—particularly given New York taxi drivers—so the ability to continually change streets can be extremely valuable. Thus, even at the more abstract level, a reactive system may be the right type of controller.

4.3 Treacherous paths: (projection, projection)

On the other hand, consider a problem such as rock climbing or walking across a rocky shore. Here we also have a high level/low level structure: you can see the terrain from a distance and plan out a coarse path, but you can't plan the details of your movements until you're close enough to see the handholds and footholds. You may well want to use some sort of projection for high level path selection, be it through classical planning or not. However, in these environments, even *following* the paths may require large amounts of projection, even if the actual selection of handholds and footholds is still highly opportunistic. Beginning rock climbers in particular do a great deal of explicit reasoning, like "if I go for that hold with my left hand, then I'll stretch my right leg too much and I might fall."

4.4 Weaving through traffic: (reaction, projection)

One can even imagine odd situations which call for reactive selection of paths, but deliberative following of paths. Imagine trying to do optimal weaving through traffic in Manhattan. You might want to do elaborate projection to decide when it's safe to perform a given illegal passing operation. You may need to plan out the timing in advance ("wait to start until the blue Honda passes, but finish the pass before we get to the intersection"). Thus the "low level" path following might be arbitrarily complicated. On the other hand, there's no point to committing to an extended series of streets in advance, for the same reasons cited above.

5 Environmental Variations

In short, environments and actions vary along several dimensions. The location of a particular environment and action model within this space partially determines which results will be optimal according to a variety of metrics.

A *predictable* environment is one in which changes can be reliably anticipated. This may be because the environment does not change—as in the original single-agent benign-environment planner model—or because it changes only in well-understood ways, as in simple linear systems. Note that predictability does *not* imply a static environment. For example, the video game Pac-Man has a predictable but rapidly changing environment: the rules governing the paths of the ghosts are extremely simple, and expert players frequently use precompiled plans for particular situations. Strategy games such as chess are at the opposite end of the spectrum: their inherent unpredictability means that prior projections will generally be invalidated by the next move, making commitment unwise. In a predictable environment, it is safe to commit to your projections (or even to precompile them); in an unpredictable environment, commitment—and often even projection—is inadvisable.

A *stable* environment changes only slowly with respect to the rate of actions and action selection of the agent. In one extreme, a stable environment does not change at all (other than by action of the agent); at the other, waiting even briefly may dramatically alter the environment in which the agent finds itself. Stable environments are predictable, allowing both projection and commitment; unstable environments may be either predictably or unpredictably so.

A *hostile* environment is one in which certain actions (or lack thereof) can be extremely costly. In these environments, when stable, experimentation is discouraged and added time to project may have a substantial payoff. When unstable and predictable, precompiled plans (advance projection coupled with real-time reaction) can be invaluable. Unstable, unpredictable, and hostile environments are among the most difficult to deal with: rapid (unpremeditated) action can be dangerous or, alternatively, necessary.

A *complex* environment is one in which projection is costly or difficult. Environments may be complex because they are unpredictable, or simply because they are difficult to characterize. Instability and unpredictability contribute to environmental complexity, but so does a wide range of variation in environmental circumstance or a dependence on hidden variables in the environmental state. Given the complexity of projection, off-line projection (as in universal plans or fixed action patterns) may be extremely valuable. Conversely, simple environments increase the utility of on-line real-time projection and, often, commitment.

An action is *reversible* if its effects can be undone at cost not substantially greater than the original action. For example, navigation in a benign environment is generally reversible. When most actions are reversible, experimentation becomes cheap(er) and the relative value of projection—mental experimentation in place of physical—decreases.

6 Interesting intermediate points

Of course the world never was divided into "pure planning" and "pure reactive" systems. There is a long history of interesting intermediate points which perform varying degrees of projection and commitment without simply nailing STRIPS onto a subsumption program.

The spreading-activation based systems of Mataric's Toto robot [15] and Maes's behavior networks [14] are both examples of what amount to classical planners which have been transformed into reactive systems by constantly rerunning the graph search (c.f. section 3). They perform projection with minimal commitment. (Toto's "planner" sits atop a more traditional reactive navigation system.)

Many reactive systems are generated some sort of deliberative system (*e.g.* a compiler or planner). In such cases, we can think of the reactive system as utilizing projection that was performed "off-line" by the deliberative system. The most extreme case of this is Schoppers' universal planner [18], which exhaustively searches the space of possible situations so as to generate a situation→action table that can drive a reactive system. Kaelbling's GAPPS system [11] compiles abstract goal specifications directly into sequential circuit descriptions. Stein's MetaToto system [19] performs additional off-line projection to the Toto system by explicitly simulating its own reactive control system in a metric model of the environment provided by an operator.

Many systems "cache" on-line deliberation into large-scale chunks that can be used in the future. Thus on-line projection is reused as off-line projection. The earliest example of this was the MACROPs system used on Shakey the robot (see Fikes, Hart and Nilsson [6]), which used a form of explanation-based generalization to generalize and store plans for future use. Sussman's HACKER also maintained a libraries of plans (programs), potential bugs, patches to gradually improve its performance over time. SOAR (Laird, Newell, and Rosenbloom [13]) uses chunking of old reasoning as a general model of learning and performance improvement. Hammond's CHEF system [8], which uses an extensive database of previous cases to do as little new planning as possible.

The extreme case of the caching approach is to store the cached projection in the form of reactive "hardware" which has direct access to the sensors and effectors. Agre's Life system [2] used a conventional rule-based system to direct action in a simulated blocks world, but cached the results of the rule firings in a truth maintenance system. The system ran very fast in familiar situations, but could handle "novel" situations by slowing down. Agre and Chapman's Pengi system [3] was effectively Life with the rule-based system removed and the TMS network compiled by a programmer. Nilsson's Teleo-reactive programming system [17] also incrementally compiles to something similar to combinatorial logic, but provides the programmer with a full lisp-like language for specifying programs.

Finally, there are examples of system that use reactive systems for both high level and low level operations. The Polly robot (Horswill [10][9]) is a running example of the reactive/reactive architecture for grid following discussed in section 4.2. Agre and Horswill's Toast system [1] handles subgoaling and coordination of multiple goals in a purely reactive system by storing the "planner state" out in the world. Of course, it can only do this because it knows a lot about the particular class of problems on which it works (cooking in a simulated kitchen).

7 Summary

The point of the foregoing discussion is not to advocate reaction over planning, or even to advocate any particular architecture at all. Rather, we would like to decouple projection from commitment from abstraction and to encourage use of a wider range of points in the projection/commitment/abstraction space. In our own work, we have often found non-traditional approaches to be surprisingly appropriate. The question of what approach is best suited to a particular environment and task can ultimately only be determined by careful study of both. We hope that this paper will encourage others to explore more versatile architectures than the TLA.

References

[1] Philip Agre and Ian Horswill. Cultural support for improvisation. In *Tenth National Conference on Artificial Intelligence*, Cambridge, MA, 1992. American Assoiciation for Artificial Intelligence, MIT Press.

[2] Philip E. Agre. The dynamic structure of everyday life. Technical Report 1085, Massachusetts Institute of Technology Artificial Intelligence Laboratory, October 1988.

[3] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, 1987.

[4] Jonathan H. Connell. Extending the navigation metaphor to other domains. In Marc Slack and Erann Gat, editors, *Working notes of the AAAI Fall Symposium on Applications of Artificial Intelligence to Real-World Autonomous Mobile Robots*, pages 29–35, Cambridge, Massachusetts, 1992. American Association for Artificial Intelligence.

[5] Irenaus Eibl-Eibesfeldt. *Human Ethology.* Aldine de Gruyer, New York, 1989.

[6] Richard Fikes, Peter Hart, and Nils Nilsson. Learning and executing generalized robot plans. In Bonnie Lynn Webber and Nils J. Nilsson, editors, *Readings in Artificial Intelligence*, pages 231–249. Morgan Kaufman, Los Altos, CA, 1981.

[7] Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings, AAAI-92*, 1992.

[8] Kristian J. Hammond. Chef. In Christopher K. Riesbeck and Roger C. Schank, editors, *Inside Case-Based Reasoning*, chapter 6, pages 165–212. Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[9] Ian Horswill. Polly: A vision-based artificial agent. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 824–829. AAAI, MIT Press, 1993.

[10] Ian Horswill. *Specialization of perceptual processes.* PhD thesis, Massachusetts Institute of Technology, Cambridge, May 1993.

[11] Leslie P. Kaelbling. Goals as parallel program spcifications. In *Proceedings, AAAI-88*, pages 60–65, St. Paul, MN, 1988.

[12] Benjamin J. Kuipers and Yung-Tai Byun. A robust, qualitative approach to a spatial learning mobile robot. In *SPIE Advances in Intelligent Robotics Systems*, November 1988.

[13] John E. Laird, Allen Newell, and Paul S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[14] Pattie Maes. How to do the right thing. AI Memo 1180, MIT Artificial Intelligence Laboratory, December 1989.

[15] Maja Mataric. A distributed model for mobile robot environment-learning and navigation. Technical Report 1228, Massachusetts Institute of Technology, Artificial Intelligence Lab, May 1990.

[16] David McFarland. *Animal Behavior: Psychobiology, Ethology, and Evolution.* The Benjamin/Cummings Publishign Company, Inc., Menlo Park, California, 1985.

[17] Nils J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1994.

[18] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings 10th IJCAI*, pages 1039–1046, 1987.

[19] Lynn Andrea Stein. Imagination and situated cognition. *Journal of Experimental and Theoretical Artificial Intelligence*, to appear.