Evolutionary Artificial Intelligence

Nils J. Nilsson
nilsson@cs.stanford.edu
Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305 USA

The Need for Coherence

Several challenges confront the organizer of an introductory course in artificial intelligence (AI). First, one has to decide what subject matter to include. The union of everything in all of the popular AI textbooks is much too large, and the intersection undoubtedly won't include enough of what the organizer thinks important. The second challenge is how to blend the selected topics into a coherent whole. The third involves matters such as problem sets, programming exercises, laboratory work, case studies, and collateral readings. Finally, one must decide on the main purpose of the course: is it to teach AI techniques and skills, or is it to study AI's intellectual content, perhaps presenting related topics in psychology and philosophy? In this note we concentrate on the first and second of these topics—how to present a coherent view of the core subject matter of AI.

Just as Los Angeles has been called "twelve suburbs in search of a city," AI might be called "twelve topics in search of a subject." The standard topics usually include Search, Representation, Reasoning, Vision, Planning, Learning, Uncertainty, Natural Language, Robotics, Game-Playing, Expert Systems, and Lisp (or Prolog). It has been difficult to present these topics in any kind of unified fashion anchored to a central theme. I have previously tried both logic and search-based problem solving as unifying motifs, but neither has been completely successful. Yet, I think it is pedagogically very important to present AI as something more than a collection of disconnected topics.

For the last two years I have been teaching an introductory AI course that is based on a progression of ever more complex and competent *agents*. Because the progression follows what plausibly might have been milestones in the evolution of animals, I have called my approach *evolutionary artificial intelligence*, even though that phrase might risk confusion with the unrelated approach to AI based on genetic algorithms.

The Progression from Simple to More Complex Agents

All agent-oriented treatments of AI, including the one I describe here, must decide what sort of entities can be called "agents." Certainly something as complex as HAL 9000 of science fiction fame qualifies as an agent, but what about much simpler things such as thermostats? In my course, I start with the very simplest kinds of agents and gradually work my way up toward much more complex ones. (When do we cross the threshold into AI? You tell me!) Here is how the progression goes:

Reactive Agents

The simplest agents merely react to their inputs by producing an output. They can be modeled by a mathematical function, $f(\mathbf{X})$, where $\mathbf{X}$ is a vector of sensory percepts and f is the output. Typical examples are phototaxic robots whose forward motion is controlled by comparing the outputs of two sensors measuring the strength of a light source. More complex reactive agents are achieved with more complex functions and percepts. Even at this early stage, we can introduce pattern recognizers and some topics in machine vision. We can also describe recently popular AI approaches such as the subsumption architecture and its variants.

Next we describe learning procedures for inducing the function, f, given a training set of inputs and their desired responses. Here, we can discuss neural nets, decision trees, and other machine induction methods. We also briefly treat genetic algorithms and genetic programming at this point.

State Machines

Using the world as its own representation works only when everything an agent needs to know at the moment of action can be immediately sensed. Otherwise, an agent needs to remember previously sensed information. We begin the treatment of representations by introducing machines that can store "state bits," simple binary-valued pieces of information that are used as part of an augmented input vector. We still have reactive machines at this stage, but now the reaction

is to the sensed information plus the state bits, and the output can affect both the world and the value of the state bits. Agents of this type are Turing equivalent (if we have an unbounded number of state bits), so further elaborations just make them more efficient.

A more complex form of representation than state bits is an *iconic* one. For example, a data structure representing a map of a robot's environment is an iconic representation. In order to get information out of a map (such as is there a doorway connecting rooms 37 and 38?), routines similar to those used for perceptual processing are needed. (I'm not sure how precise I can make the distinction between iconic and non-iconic representations, but I think the intuition is right.) My agents can now base their actions on sensory information, state bits, and on information derived from their iconic representations. Their actions might also make changes to the iconic representations.

Planning What to Do

There are many ways to implement the input-output function, f. An agent might have an explicit table listing the appropriate reaction to every possible input. Presumably the agent's designer computed the table at *design time*. An alternative is to have the agent do the computation at *run time*. For many applications, neither the table nor a simple functional form is available. What might be available, though, is an *action model* that tells the agent what its next input (sensory plus state bits) would be for any possible action that it might take, given its current input. Such an agent can compute an action by trying out various ones in its model and evaluating their consequences. Thus, we come to the various search procedures important in AI—depth-first, breadth-first, A*, and so on. We also take a slight detour here to talk about game-tree or adversarial search.

According to some authors, an agent becomes *autonomous* only when it evaluates the consequences of its actions. Of course, we have no way of knowing from its external behavior whether an agent chooses an action by referring to a table or by a search process—the matter is purely one of implementational efficiency and space-time trade-off. Nevertheless, we think of agents that plan what to do as being more sophisticated than those that *merely react.*

Logical Agents

In addition to having some computational means to compute output from input, an agent may know that there are certain constraints or dependencies governing its input. In the simple case in which the input is a vector, $\mathbf{X}$, it may be that certain vectors are impossible; or, given the values of some of the components, the agent may be able to compute the values of others. What is the appropriate language for describing these constraints and dependencies? Propositional calculus, of course! Here we lay the groundwork for logical representation and reasoning—introducing resolution and the various proof strategies that use resolution. My agents are now capable of elementary logical reasoning, and we can describe simple expert systems at this stage of the course.

Moving toward a more expressive language for representing constraints, we next introduce first-order predicate calculus including, unification and first-order resolution. This point also affords us the opportunity to introduce Horn clauses, logic programming, and the trade-offs between expressibility, soundness, completeness, and computational feasibility. Then, we move on to more structured representations such as frames and semantic networks. Nonmonotonic reasoning is introduced as a formalization of inheritance cancellation in semantic networks.

Of course, the formalism in which knowledge is represented doesn't tell us anything about *what* ought to be represented. The exercise of attempting to represent commonsense as well as expert knowledge gives us a flavor of the problems involved in constructing ontologies.

We confront the fact that much useful information is uncertain by introducing techniques associated with probabilistic logic, belief networks, and other related formalisms. More complex expert systems using these methods can then be presented.

Planning Using Logical Descriptions

Having a language in which to *describe* the effects of actions (instead of having to model them iconically as we did when we discussed search methods earlier) permits more powerful planning methods. These methods, based on the situation calculus and STRIPS rules, leave some matters about the effects of actions unspecified, and that is both good and bad. The bad part is manifested in the frame problem (and its friends). We present a progression of planning techniques—culminating in non-linear and hierarchical planning. Our agents are now capable of rather sophisticated problem-solving behavior, but the computationally feasible ones are prone to various shortcomings associated with the methods they use.

Communication and Multiple Agents

Agents must communicate with humans and other agents. Although there is little material dealing with specialized agent-to-agent communication languages, we can discuss here the concepts of speech acts in the context of planning to affect the cognitive structures of other agents. These topics underlie our treatment of natural language processing, which is dealt with next.

Integration and Agent Architectures

So far we have described agents that react, agents that reason, agents that plan, and agents that communicate. Now we integrate these abilities in robots

and softbots that are much more capable than the purely reactive agents considered at the beginning of the course. Various architectures capable of real-time deliberation are presented.

Summary and Conclusions

My evolutionary approach helps tie together aspects of AI that might otherwise seem disparate. Student reaction has been positive. Programming assignments allow students to build their own progression of agents. Stanford uses the ten-week quarter system, but I think the course would be better matched to a fifteen-week semester.

Some people have criticized an agent-oriented approach to AI—saying there is more to AI than agents. Perhaps so, but I think that some of these other aspects of AI, such as expert scheduling systems, design systems, and diagnosis systems are a bit off the track of what should be the main goal of AI—developing versatile agents that can use these other systems as tools. In any case, the material presented in my course is pretty much what ought to be in a *first* course in AI.

It is a feature of my approach that machine learning is not treated as a separate topic; instead, various aspects of learning arise throughout the course. Neural nets can be presented early; techniques for learning search heuristics can be discussed while talking about search; inductive logic programming and explanation-based learning can be treated toward the end of the logic block, and learning search control knowledge can be presented after discussing logic-based planning.

I am intending to transform my rough lecture notes for this course into an AI text. Interested people can contact me in late 1995 for early draft material.