Learning intermediate goals and methods from situated instructions

Scott B. Huffman
Price Waterhouse Technology Centre
68 Willow Road
Menlo Park, CA 94025
huffman@tc.pw.com

Introduction

Despite many recent advances in knowledge acquisition tools and techniques, acquiring knowledge is still a major bottleneck in building intelligent systems for complex domains. Typically, a team of knowledge engineers extensively interviews domain experts, and then manually translates the information obtained into a form that the intelligent system can use. This process is costly and time consuming. In addition, all of the expert knowledge must be put into the system a-priori; after the system is running, human expertise cannot be used to change or add knowledge (short of turning off the system and reprogramming it).

In this work, we are exploring an alternative: learning domain knowledge from *situated instruction* given by the domain expert. In situated instruction, an instructor describes and explains appropriate goals and actions in specific situations – either the current situation, or one set up by the instructor. From these specific instructions, the system will learn general knowledge about goals and methods that can be used for planning and acting in the domain. Instruction allows the system to take advantage of human expertise during its ongoing operation.

We believe that instruction is a potentially powerful knowledge source for systems learning to plan and act in complex domains. From instruction, a system can learn completely new goals and methods [Huffman, 1994]. Because of the large space of possible goals and methods in complex domains, it would be difficult for a system to "discover" many of the useful goals and methods without human guidance. Indeed, learning techniques that do not involve human guidance, like exploration and experimentation in the environment (e.g., [Gil, 1992; Shen and Simon, 1989]) have concentrated on revising knowledge about domain operators, rather than discovering new performance goals (although LIVE [Shen and Simon, 1989] can discover hidden features in its environment that affect performance).

This paper describes techniques we have developed for learning intermediate goals and associated methods from interactive natural language instructions. The techniques are implemented in an agent called Instructo-Soar [Huffman, 1994; Huffman and Laird, 1994]. To date, Instructo-Soar has been applied in depth only to a small, simulated robotic domain (not a "real" domain). Truly flexible instructability in real domains makes a number of difficult demands on an instructable agent, that we have enumerated elsewhere [Huffman, 1994]. In this paper, after briefly illustrating Instructo-Soar's techniques with examples from its toy domain, we turn to a protocol of human instruction in a real domain (tactical air combat). This protocol is used to highlight a number of extensions required to move our instructable agent closer to real domains.

Intermediate goals and methods

A key type of knowledge that can be learned from instructions is intermediate goals and associated methods for achieving them. Inspired by our tactical-air protocol, we call these intermediate goals/methods *tactics*. Tactics involve sequences of actions to achieve some goal that is appropriate in particular situations. Thus, to learn a tactic, and agent must learn a *new goal*, *how* to achieve that goal (what actions to use, possibly in different situations), and *when* to use the tactic (when the goal should be attempted).

Tactics differ from the highest-level goals of the agent (e.g., survival, conserving resources), which are simply given to the agent at its outset; and they differ low-level knowledge of basic domain actions (e.g., opening/closing a robot's gripper, manipulating a plane's controls). They are intermediate goals/methods, constructed out of low level actions, to perform in service of higher level goals.

An example of a tactic Instructo-Soar learns from instruction is the *dim-lights* tactic: it learns how to dim a light, and when and why this should be attempted. In our tactical-air protocol, a flight instructor describes a tactic called the *spacing maneuver* that a pair of planes uses to gain a tactical advantage over a single fighter.

Why use instructions?

Why use instructions to learn a tactic? For an agent to discover useful tactics without instruction, it would

have to search in the space of possible goals/methods (a huge space in a domain of any complexity), and determine which are most useful for achieving higher-level goals. Each goal/method combination would have to be evaluated in a large number of particular situations, since tactics include knowledge indicating in what situations they should be used. Human expertise in many domains (e.g., air combat) is the result of many years of searching for useful tactics. It would be unproductive to ask an AI agent to recreate those years of search to discover tactics without human guidance.

Systems that learn from different types of expert guidance fall under the rubric of *learning apprentice systems* (LAS's) [Mitchell *et al.*, 1990]. LAS's typically learn either by observing expert actions (e.g., [Mitchell *et al.*, 1990; Wilkins, 1990], or attempting to solve problems and allowing the expert to critique decisions as they are made (e.g., [Laird *et al.*, 1990; Kodratoff and Tecuci, 1987]). Although LAS's have been applied to many tasks to learn different kinds of knowledge, they typically do not learn new goals, but rather let the instructor choose from known actions at choice points. Also, past LAS's do not handle the situated and explanatory types of instructions that instructors often use to teach tactics (described below). Instead, most LAS's have focussed on learning from observing/performing actions non-intrusively, as opposed to actively seeking instruction.

Some programming-by-demonstration systems (e.g., Eager [Cypher, 1993, ch. 9]) learn what are essentially new goals by inferring procedures for oft-repeated sequences of user steps. This kind of observational learning is an alternative to the verbal instruction approach we are using here (or could be combined with it), but again, in a complex domain it may be difficult to infer useful goals from observation alone.

Approach

Instructo-Soar learns hierarchies of new goals/methods and other kinds of domain knowledge (inferences, control knowledge, effects of actions) from instructions. In its simulated robotic domain, it learns goals like picking up and lining up objects, how to control simple devices, etc., and domain knowledge like the effects of a magnet on metal objects. An important aspect of Instructo-Soar is its attempt to allow highly flexible instruction. In particular, it allows the instructor to request unknown or multiple-step actions at any point in the instruction dialogue (as opposed to only directly executable actions), and to use instructions that explicitly indicate the situation they apply to (e.g., conditionals).

Instructo-Soar uses a method called *situated explanation* to learn from instructions. Briefly, for each instruction the agent is given, it first determines what *situation* the instruction applies to (either the current situation or a hypothetical one indicated by the language of the instruction). A situation consists of a state, and a goal that the instruction leads towards (or precludes, for negative instructions) from that state. Within the instruction's situation, the agent next attempts to *explain* why the instruction allows goals to be achieved in that situation. If this explanation succeeds, general learning results, a-la EBL [Mitchell *et al.*, 1986; Rosenbloom and Laird, 1986]. If an explanation cannot be completed, it indicates that the agent lacks some knowledge needed to explain the instruction. This lack is dealt with in different ways depending on the instructional context, as described below.

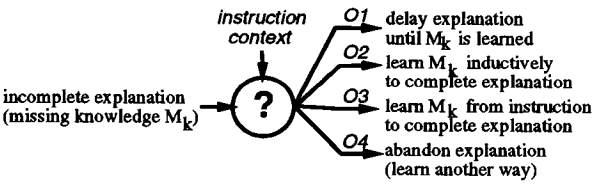

Figure 1: Options when faced with an incomplete explanation because of missing knowledge M_K.

In Instructo-Soar, explanation takes the form of a *forward internal projection:* the agent internally places itself in the situation state, and then runs forward, applying the instructed step, and knowledge that it has about subsequent states and actions. If the situation goal is reached within this projection, then the projected path through the instructed step comprises an an explanation of the instruction.

However, the agent's knowledge is often incomplete, precluding the formation of an explanation. For instance, the agent may not know a key effect of an operator, or a crucial state inference, needed to reach the goal within a projection.

There are a number of general options any learning system might follow when it cannot complete an explanation. (O1) It could delay the explanation until later, in hopes that the missing knowledge will be learned in the meantime. Or, (O2-O3) it could try to complete the explanation now, by learning the missing knowledge somehow. The missing knowledge could be learned (O2) inductively (e.g. by inducing over the "gap" in the explanation, as in [VanLehn *et al.*, 1992] and many others), or, (O3) in an instructable agent's case, through further instruction. Finally, (O4) it could abandon the explanation altogether, and try to induce the desired knowledge instead.

Given only an incomplete explanation, it would be difficult to choose which option to follow. Identifying missing knowledge in general is a difficult credit assignment problem, and nothing in the incomplete explanation predicts whether this knowledge will be learned later if explanation is delayed. Thus, most machine learning systems either follow a single option (e.g., [VanLehn *et al.*, 1992] follows O2) or statically order options (e.g., OCCAM [Pazzani, 1991] tries O2 and then O4).

```
> Push the red button.
How do I do that?
    > Move to the grey table.
    > Move the hand above the button.
    > Move down.
    > The operator is finished.
    Does "push the button" mean having
    your hand directly above the button?
    > Right.
    ...
> To dim the light, push the green button.
    Does this mean that an effect of pushing the
    green button is that the light is now dim?
    > Yes.
    ...
> If the light is bright then dim the light.
Can you tell me why?
    > Bright lights use much electricity.
    > Dim lights use little electricity.
    Gotcha.
```

Figure 2: Instructions for the *dim-lights* tactic.

However, as indicated in Figure 1, an instructable agent can use additional information about the instructional context (that is, the type of instruction and its place in the dialogue) to determine which option is most appropriate for a given incomplete explanation. In order to cover a diversity of instructional situations, Instructo-Soar uses all four of the options O1-O4 under different circumstances [Huffman and Laird, 1994].

Example: Learning the *dim-lights* tactic

One set of instructions for teaching Instructo-Soar a tactic is shown in Figure 2. The first series of commands teaches the agent a new goal (pushing a button) and a method (a sequence of steps) for achieving it. Since the goal is unknown initially, the agent delays explanation of the steps. When the instructor indicates that the goal is met, the agent uses a simple "differences" heuristic to induce the conditions under which the goal is met. Then, the agent recalls and explains each of the previous instructions to learn a general method for the goal. For instance, it generalizes the first instruction by learning to move to the table that the button is on, instead of specifically to the grey table.

Next, the instructor indicates a use for the push-button goal: dimming the light. The instruction `To dim the light, push the green button` contains a *hypothetical goal* and an action to achieve that goal. Thus, the agent creates a situation in which the light is not dim, and forward projects the action of pushing the green button within that situation. However, the agent has no knowledge that the button and the light are connected, so its projected action has no effect on the light. In this case, based on the form of the instruction there is a strong expectation that the goal *should* be met by performing push-button. Thus, the agent attempts to complete its explanation through induction. Here, it induces a new operator effect: pushing the green button makes the light become dim.

In the final series of instructions, the instructor indicates and explains a situation in which the tactic should be used. The first instruction, `If the light is bright then dim the light`, indicates a hypothetical state. The agent tries to explain to itself why the light should be dimmed in a situation where it is currently bright, but cannot. In this case, since there is no strong expectation generated by the instructional context, the agent defaults to asking for further instruction. The instructor's next two statements teach the agent about the different resource needs of bright and dim lights. These statements complete the agent's explanation of the original instruction by keying off of prior knowledge about the high-level goal of conserving resources.

This example demonstrates that Instructo-Soar's techniques can be used to learn simple tactics, involving a new goal, a method for achieving the goal, and knowledge of what situations the goal is appropriate for. How far are these techniques from application to real instruction in a real domain?

A real example: the *spacing maneuver*

To address this question, we have examined a protocol of instruction from a tactical air (fighter plane) domain. The protocol, shown in Table 1, is a one minute excerpt from an interview with an expert, in which he teaches a tactic called the *spacing maneuver*. The protocol was collected as part of a knowledge-acquisition session for constructing Soar agents that control fighter planes in real time [Jones *et al.*, 1993] within a large-scale distributed simulation environment. The domain is definitely "real" in the sense that real military personnel use the simulation for training, and the intelligent agents are expected to produce human-quality performance.

The spacing maneuver involves two planes, in a line flying straight towards their enemy; the first plane turns, and loops around behind the second, which becomes the new lead plane. The tactic builds on low level actions like turning, and is performed in service of the higher-level goal of gaining a strategic advantage over the enemy plane, in a particular type of situation.

This protocol has four important properties that force scaling of Instructo-Soar's current techniques:

1. **Multiple-utterance explicitly situated instructions.** The early instructions (1-3, 5-6) explicitly set up a hypothetical situation that the rest of the instructions apply to. Instructo-Soar's situated explanation approach deals with such explicitly situated instructions, but can only handle hypothetical situations that exist within a single utterance. The research issue in extending to these instructions is dealing with hypothetical situations that are created and exist through multiple utterances.

#	Utterance	Action
1	Sometimes, like out at 25 miles	grabs two red planes
2	These guys now, they'll come in,	lines up reds in a row
3	It'll be a trail,	
4	They know that we like to take our phoenix shots in, somewhere in the mid 20's	
5	So at about 22 miles,	starts to turn lead red guy
6	thinking that they will have already committed a missile to this guy	taps lead red guy
7	this guy...they'll basically just change places	turns lead red behind, and moves rear red forward (they change places)
8	Ok, they're doing a spacing maneuver...	picks up former lead red
9	but it makes the missile that was shot against this guy go stupid.	shakes former lead red
10	they can't aerodynamically get there, ok...	
11	It also creates some confusion	motions towards blue, then puts original lead red back and starts turn for spacing maneuver
12	because they see this guy extrapolate on the radar, ok,	
13	and then another guy, y'know, they, uh,	continues through spacing move w/reds
14	another guy pops up at about the same range	motions to new lead red
15	well, is this guy hovering?	motions to new lead red
16	or is this a different guy? what happ...?	
17	it creates some confusion on the part of the fighter	motions to blue
18	he is hesitant to quickly employ another weapon against this guy	motions to reds
19	because he doesn't want to lose another phoenix, ok?	
20	so what that ends up doing	puts reds at end of spacing maneuver
21	is now they get into where they can exchange medium-range shots	indicates distance from blue to lead red w/fingers
22	now it's a fair fight, in fact it's kind of weighted on the side of the red side, ok?	

Table 1: A portion of a knowledge-acquisition session in which the spacing maneuver is taught. This protocol lasts about one minute.

2. **Complex explanatory instructions.** In addition to the situation and the actions for the maneuver, the instructions contain explanatory material (e.g., 4, 11-19) describing the results of the tactic, and justifying its use. What do these instructions contribute to learning? At least one of their roles is to allow better *generalization* of the knowledge being learned. For example, consider instructions 4-5: "They know that we like to take our phoenix shots...in the mid 20's, so at about 22 miles..." The instruction gives a reason for action occurring "at about 22 miles". An agent could use this reason to learn something general; e.g., "Act just after you think the enemy takes missile shots" rather than simply "Act at 22 miles." The more general knowledge would apply, for instance, if the enemy had a new kind of missile that they shot from 40 miles out.

   Instructo-Soar can learn from very simple explanations of this kind, such as the explanation of why bright lights should be dimmed in the example above. However, the explanations in real instruction are more complex, and are not always initiated in the same way (e.g. by the agent asking "Why?"), so explanatory information must be actively identified.

3. **Multiple agents.** The explanatory instructions are complex in part because they apply to multiple agents. More than this, they seem to refer to these agents' reasoning processes about one another (e.g., 6, 11).

4. **Difficult language.** Finally, real instruction like that in this protocol contains very complex language. Handling the full range of instructional language seems beyond the reach of current natural language systems.

   However, we believe that within specific domains, it is possible to support communication of a wide range of instructional information through stylized graphical interfaces (buttons, lists, tables, etc). To insure the needed flexibility in this type of interface, we plan to identify both the semantic features in a domain, and the discourse-level roles and relationships of the instructions. The features will

be domain-specific (e.g., `distance-to-contact(22 miles)`). However, the relationships and roles of the different instructions are general. These might include `describe-situation` (e.g, instructions 1-3), `expected-result` (instructions 9, 11), `reason-for-result` (instruction 10), etc. Work on discourse relations (e.g., [Mann and Thompson, 1988]) may provide guidance in identifying these roles and relationships.

Conclusion and current work

We believe situated instruction has great potential as a knowledge source for an agent that must exist over a long period of time and be applied to a wide variety of tasks in its lifetime. Rather than attempting to learn everything about the world from scratch, instruction builds on knowledge of both low-level actions that the agent can perform, and high-level goals that the agent is endowed with. Instructo-Soar's learning approach reflects this, using as its core an analytic method (EBL) that makes maximal use of prior knowledge, and responding in a variety of ways depending on context when prior knowledge is insufficient. The tactics (intermediate goals/methods) that are learned are useful to the agent in future situations, both for acting appropriately and for planning.

Recently at Price Waterhouse Technology Centre, we have begun to look at instruction for an agent that aids business consultants in benchmarking the financial performance of a corporation. The task involves integrating large amounts of data from heterogeneous sources, and performing a variety of categorizations, calculations, and displays of that data. It requires an agent with a basic set of primitive actions (e.g., simple querying of databases) that can be applied flexibly to a large and changing variety of tasks, because different benchmarking practices are followed in different industries. This domain will involve extensions to Instructo-Soar related to those derived from the tactical-air protocol. We described that protocol instead of one from the financial domain because collecting a protocol for the financial benchmarking task requires building a prototype data-integration system, that is still under construction.

Acknowledgments

This research was supported in part at the University of Michigan by contract N00014-92-K-2015 from the Advanced Systems Technology Office of the Advanced Research Projects Agency and the Naval Research Laboratory.

References

[Cypher, 1993] A. Cypher, editor. *Watch what I do.* MIT Press, 1993.

[Gil, 1992] Y. Gil. *Acquiring domain knowledge for planning by experimentation.* PhD thesis, Carnegie Mellon Univ., Sch. of Computer Science, 1992.

[Huffman and Laird, 1994] S. Huffman and J. Laird. Learning from highly flexible tutorial instruction. In *AAAI-94*, pages 506–512, 1994.

[Huffman, 1994] S. Huffman. *Instructable autonomous agents.* PhD thesis, Univ. of Michigan, Dept. of Electrical Engineering and Computer Science, 1994.

[Jones *et al.*, 1993] R. Jones, M. Tambe, J. Laird, and P. Rosenbloom. Intelligent automated agents for flight training simulators. In *Proc. 3rd Conf. on Computer Generated Forces*, pages 33–42, 1993.

[Kodratoff and Tecuci, 1987] Y. Kodratoff and G. Tecuci. Techniques of design and DISCIPLE learning apprentice. *Int'l. J. of Expert Systems*, 1(1):39–66, 1987.

[Laird *et al.*, 1990] J. Laird, M. Hucka, E. Yager, and C. Tuck. Correcting and extending domain knowledge using outside guidance. In *Proc. ML-90*, 1990.

[Mann and Thompson, 1988] W. Mann and S. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

[Mitchell *et al.*, 1986] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1, 1986.

[Mitchell *et al.*, 1990] T. Mitchell, S. Mahadevan, and L. Steinberg. LEAP: A learning apprentice system for VLSI design. In *Machine Learning: An artificial intelligence approach, Vol. III.* Morgan Kaufmann, 1990.

[Pazzani, 1991] M. Pazzani. Learning to predict and explain: An integration of similarity-based, theory driven, and explanation-based learning. *J. of the Learning Sciences*, 1(2):153–199, 1991.

[Rosenbloom and Laird, 1986] P. Rosenbloom and J. Laird. Mapping explanation-based generalization onto Soar. In *AAAI-86*, pages 561–567, 1986.

[Shen and Simon, 1989] W. Shen and H. Simon. Rule creation and rule learning through environmental exploration. In *IJCAI-89*, pages 675–680, 1989.

[VanLehn *et al.*, 1992] K. VanLehn, R. Jones, and M. Chi. A model of the self-explanation effect. *J. of the learning sciences*, 2(1):1–59, 1992.

[Wilkins, 1990] D. Wilkins. Knowledge base refinement as improving an incomplete and incorrect domain theory. In Y. Kodratoff and R. S. Michalski, editors, *Machine Learning: An AI Approach, Volume III*, pages 493–514. 1990.