Tinker: Example-Based Programming for Artificial Intelligence

Henry Lieberman

Artificial Intelligence Laboratory
and Laboratory for Computer Science
Massachusetts Institute of Technology

I will show a demonstration of Tinker, an experimental interactive programming environment for Lisp, designed to support a programming methodology suited for Artificial Intelligence applications.

How does AI programming differ from areas like business or scientific programming? One important difference is in the complexity of the problems and procedures necessary for solutions. In many other areas, programming proceeds by first creating complete specifications for the problem, inventing an algorithm guaranteed to meet the specifications, then working out the details of the code, and finally, a testing phase when all the previous phases have been completed.

AI tasks are usually so complex that this methodology breaks down. Suppose I wanted to construct a natural language understanding program. There's virtually no hope of being able to write specifications or invent algorithms which completely characterize the problem before beginning coding and testing

Instead, my approach would be driven largely by considering *examples.* Usually, the AI programmer has in mind a *scenario,* a sample of the kind of interaction which will be possible with the finished system. For a natural language program, the scenario would contain examples of sentences or stories I would expect the program to understand. I would reflect on what knowledge is necessary to understand the examples, and carefully test any proposed theory on the examples. The program would develop by *abstraction* from the procedures which handle the examples. Further work proceeds by incrementally adding new cases, *generalizing* the results of each attempt.

Tinker is my first attempt at building a complete programming environment which takes this example-based methodology seriously. When you want to define a new procedure, you present examples of the data on which the procedure operates. As you perform each step of the procedure, Tinker shows what happens on the particular examples, and remembers the steps to construct a generalized procedure. A set of several examples can be given, each illustrating an important case for the resulting program. With Tinker, testing a program happens while you write it, rather than afterward.

Tinker does *not* do what is sometimes called *programming by example,* in the sense of trying to *guess* what the definition of a procedure is from a presentation of the input-output history of the procedure. Currently, the user must direct Tinker in performing each step of the procedure. Tinker's value lies in showing the programmer the results of all the intermediate steps on examples, making it much easier to detect bugs and understand the program's performance. Tinker also has the potential for supplying a friendly user interface for program understanding systems and intelligent program editors.

Tinker is implemented on the MIT Lisp Machine, a personal computer with a high-resolution graphics display and pointing device Tinker takes advantage of these capabilities to present a convenient interface to the user. Multiple windows display a snapshot of the computation, source code for programs, typed input, error messages, printed output, and graphics. Tinker interacts with the user primarily by using *menus,* requiring much less typing than conventional systems.

(Supported by Own contact W00014-76-C-0522. and DARPA contract N0O014-•0-C-0505.)