Introduction

Imagine for a moment that you’ve been assigned the task of developing and teaching a course in Artificial Intelligence (Reviewers with a background or expertise in AI should imagine themselves assigned to teach an utterly different course for which they have little background for the sake of argument) to upper-division computer science undergraduates at a small liberal arts college. You begin by visiting popular publishers’ web sites to locate potential texts and take an informal survey of course-syllabi posted on the web sites of other schools to see what textbooks are commonly used and the overall structure and content of a typical AI course. You then spend a few days perusing exam-copies of a number of texts and finally make a selection.

Over the summer you begin preparing for your new AI course. The textbook turns out to have been a good choice, as it focuses on elements of “symbolic AI” that you feel are especially relevant to undergraduates. As with any course, you begin considering ways to make the abstract concepts concrete through relevant hands-on course-work. But here is where the difficulty begins. The authors of your main text, having dedicated themselves to the study and application of AI, describe their rich environment for experimentation that involves a wonderful robotics laboratory replete with a bevy of capable graduate assistants. Another author describes a sophisticated reasoning and representation system that can be downloaded for use with assignments described in their text. You page through other textbooks and find that those authors are likewise well-endowed with not only a rich infrastructure and set of tools, but also a large faculty and a much more comprehensive curriculum than your one-semester AI course. You discover that - unlike Database Management courses where virtually every textbook and course syllabus includes a database design and implementation project - AI is a wide-ranging discipline with no consistent pedagogy for demonstrating core concepts. Furthermore, you find that teaching faculty from other similar institutions are struggling to integrate the topic of “Intelligent Systems” as proscribed by the ACM Computing Curricula 2001.

You then realize that the experience your students will receive is likely to be quite different from that described in the textbooks you have read and, more to the point, entirely up to you. And so you begin to put together a smattering of simple and disconnected programming assignments that provide an uninspiring introduction to various AI concepts.

What if you instead could provide a concrete introduction to the core concepts of AI using a motivating example that can be completed within the constraints of a one-semester introductory course in a small college liberal arts environment?

Pilot Project

Faced with this dilemma, we developed a simple but motivating AI project we call Glomus (Glomus is a rough Latin translation of the word “Clue”). Glomus addresses the core elements of “symbolic AI”, including representation and reasoning, along with a contemporary focus on “agents in action.” Glomus is based on the popular board-game Clue where the “who, where, how” of an event must be discovered. Each “agent” has limited knowledge of the state of the world and must strategically set out to discover additional information that will allow the agent to solve the mystery (i.e., complete a logical proof). Basing the project on a popular board game means that students are more likely to already be familiar with the task and the rules.

The project progressed in phases that roughly correspond to our chosen textbook where the authors suggest that computationally intelligent agents must deal with four common tasks: Modeling the environment, evidential reasoning or perception, taking action, and learning from past experience.

Students began by learning how to model the environment using PROLOG. Various ways to represent and encode facts, rules, and relationships were explored and discussed. This task provided a hands-on example that was very complementary to our textbook author’s example of situated robots as well as similar examples used by other popular textbooks. Although our students use PROLOG, the project itself is completely language-independent.

Rules of inference and proof-procedures were covered next. Students added basic rules that encoded a proof of “whodunit” - a task that was surprisingly valuable as it gave students a concrete understanding of resolution theorem-proving and the power (and limitations) of PROLOG’s inference engine.
The project then moved on to taking action. Here, the exploration of “intelligence” began as students developed novel rules and strategies for moving and making suggestions about “whodunit” based on their agent’s limited knowledge. Agents were augmented with the ability to add new facts as they “heard” what other agents had suggested and what refutations had been made. This enabled students to implement rules accommodating the evolution of knowledge so that their agent’s behavior was incrementally refined by past experience. Advanced students also took into consideration various ways our textbook presented for assumption-based reasoning and reasoning under uncertainty.

By the conclusion of the course, each student had developed a complete agent in PROLOG that was capable of playing a legal and intelligent game of Glomus. Most of these agents consisted of 1,000 - 3,000 lines of commented PROLOG code; my own highly-refined agent is comprised of just over 1,000 lines.

Students tested their agents against one another in a brief round-robin tournament. Students manually entered the start-state of the game into their agent and continued to manually update their agent with the moves and suggestions of others. When it came their turn, students would ask their agent for their move and suggestion and then verbally communicate that to the others who manually updated their agents.

While the course project has exceeded our expectations by making the tasks common to intelligent agents concrete for our students in a one-semester course, the results of their work were difficult to test due to the manual interaction required. Thus the goals and objectives for this proposal are twofold: (1) to implement a simple centralized server in software that would allow students to test their agents and compete against one another, and (2) to do so in such a way as to make the entire project accessible to other teaching faculty by providing production-quality software and extensive documentation.

Impact

This project targets “teaching faculty” at small institutions who typically teach many more courses than faculty at “research” Universities (For example, I teach 10 different courses in a two-year rotation). Teaching faculty have the need for relatively complete and easily adopted course-projects.

First, we propose to develop software that will provide a test-bed for development as well as a platform for automated competition. Currently, manual data-entry and the speed and accuracy with which students can type information into their agents is a severe limitation that results in: little or no opportunity for students to test their agents; games taking 15 minutes or more to play because the students’ time is dominated by manual data-entry; students having to forfeit on occasion due to inaccurate data-entry, making both testing and tournament play tedious and frustrating; and “Luck” playing a disproportionate role in the outcome of the final competition due to the small number of games students were able to play.

The server software we develop will enable incremental testing as students test their agents against existing agents accessible to the server. This will allow students to both debug their code as well as test and evaluate different heuristics; eliminate manual data-entry time and errors; and facilitate automated tournaments consisting of many games that will amortize “luck” and give a more accurate assessment of individual student achievement - therefore allowing faculty to recognize and reward those students whose agents are incrementally better than others.

The server will not only greatly enhance the project for our course but also will help to fulfill our second objective - making an entire self-contained project accessible to other teaching faculty. The server software will be a self-contained system of software utilizing a simple language-independent communication protocol allowing others to develop client agents in any language. Faculty thus will be able to easily adopt the project for their own AI courses, focusing on the core objectives of AI without having to learn a particular language, develop expertise in a specialized logical-reasoning framework, or acquire and assemble hardware. Client agents can be coded in any language using any strategy deemed fit.

With the funds received from the NorthWest Academic Computing Consortium we will also develop curricular materials that describe the Glomus project, including the server and client agent implementation along with example agents coded in various languages. These curricular materials, together with a self-contained distribution of the server software, will be made available to faculty who wish to adopt and adapt the project.
Project staff will include the proposing faculty member together with 3 undergraduate students, one of whom is currently funded via a Richter Scholars Grant from George Fox University

Schedule

The project will proceed in four phases: (1) server design; (2) beta implementation and documentation; (3) final implementation and testing; and (4) dissemination.

PHASE ONE: 1/2005-5/2005 — Server Software Design. The first phase will set out the design for the server software. Design issues to be addressed include an object-oriented analysis and design of the overall architecture and the server software resulting in a well-formed UML diagram that will serve as the basis for implementation and subsequent project documentation; the specification of the protocol used for client-server communication including syntax and semantics for starting games involving multiple clients, turn-taking, game resolution, record-keeping, and error-handling for common problems such as communication failure, illegal moves, and client timeout due to excessive compute-time.

This phase is already underway and we hope to have the protocol and initial design of the server finished by June of 2005.

PHASE TWO: 6/2005-8/2005 — Implementation and Project Documentation. The second phase will begin in the summer of 2005 and run through August of 2005. The goals of this phase are to implement a beta-version of the server and develop a web site providing high-quality curricular materials describing the project and its adoption for use by other faculty. Support from NWACC will fund 2 undergraduate students and one faculty member who will implement the software server and the web site. NWACC funds will also be used to establish a project web site that will describe the project and its progress as well as to provide access to the software developed together with curricular materials that can be used by other faculty to adopt this project. The web site will include mechanisms for registration so that we can follow up with faculty on their use of the project materials and garner feedback for project improvement.

PHASE THREE: 8/2005-5/2006 - Final Implementation. The software will be incrementally refined throughout the fall of 2005 with the goal of having a fully-functional production-quality server available for use during the next iteration of our AI course in the spring of 2006. Once we have successfully deployed and used our software system ourselves we will make the finished product available for wider adoption.

PHASE FOUR: - Project Dissemination and Evaluation. We have targeted two conferences for dissemination of our work. We will submit a paper and/or student posters to the Pacific Northwest region Consortium for Computing Sciences in Colleges (CCSC) Conference in the fall of 2006. Depending on the state of our progress, we may also submit a paper to the ACM Special Interest Group for Computer Science Education (SIGCSE) conference in the spring of 2005. We will also seek to have our web site referenced by others such as the “Pedagogic Resources in AI: Home Page” at Temple University and the educational web-pages at the American Association for Artificial Intelligence (AAAI). Finally, in addition to soliciting feedback from faculty who adopt the project, Professor David Poole of the University of British Columbia, a co-author of the popular AI textbook we use, has offered to provide us with a review of our project and the curricular materials we have developed.