A Repository that Supports Teaching and Cooperation in the Introductory AI Course *

Giorgio Ingargiola†, Nathan Hoskin, Robert Aiken, Rajeev Dubey, Judith Wilson
Temple University

Mary-Angela Papalaskari
Villanova University

Margaret Christensen
Drexel University

Roger Webster
Millersville University

Abstract

This paper describes the development of FLAIR (Flexible Learning with an Artificial Intelligence Repository), a repository of educational material and of a highly visual computing environment for use in laboratories associated with the introductory undergraduate Artificial Intelligence (AI) course. This repository supports sharing of pedagogic material and of development tools, and cooperation in their use, while allowing diversity in content and in use at different institutions. Thus the development of the repository has stressed the production of system tools, extensible object-oriented libraries, and strong programming frameworks. Some modules currently available are on Search and Automated Reasoning. Examples of the presentation techniques used are conceptual maps, hypertext, and graphic animations of algorithms. Initial experience in the use of the repository in teaching the introductory AI course is taking place in the 1993/94 academic year.

1 Introduction

The introductory undergraduate Artificial Intelligence (AI) course can be difficult to teach and, for students, difficult to learn. The course covers a diverse and loosely connected array of topic areas, many of which assume that students have facility in mathematics and logic. In addition, students must master new programming paradigms when they learn Lisp or Prolog, as typically required for course assignments. Computer science instructors at small or moderately-sized institutions often have not had much exposure to AI and cannot draw upon the resources of active AI research efforts at their institutions. For these reasons, the AI course is at risk of presenting general principles abstractly and superficially, or, alternatively, of becoming another programming course.

This paper describes a project intended to help faculty manage the difficulties associated with teaching the first undergraduate AI course (Aiken *et al*, 1992). [1] Our project, entitled FLAIR, "Flexible Learning with an Artificial Intelligence Repository," has as its goal the development of a repository of educational material and of a highly visual computing environment for its use in laboratories associated with undergraduate courses in AI.

In developing this repository we are guided by three principles.

- **Sharing:** The development of effective pedagogic material is difficult, time consuming, requiring AI and teaching knowledge. Thus, users of the repository would benefit from sharing material on different topics or with different interpretations of a single topic, developed by different groups.
- **Diversity:** Diversity of repository content across sites is the natural result of differences in the development and collection of material, and should be supported within the repository environment. Diversity is also necessary within a site, so that different users can add new material or annotate existing material without impeding each other.
- **Cooperation:** It is important to support cooperation among students, among students and instructors, and among instructors, both within a site and across sites. This cooperation goes beyond the sharing of available information. It involves active, directed interactions in ways that support learning.

*This research has been supported in part through the Educational Infrastructure Program of the National Science Foundation (Grant #CDA-9115254).

†Email: ingargiola@cis.temple.edu

SIGSCE 94- 3/94, Phoenix, Arizona,USA

[1] Inspiration for some aspects of this project was received from the Portable AI Lab project (National Science Foundation of Switzerland, Project NFP 23) (Aiken, Allemang, & Wehrle, 1992).

For example, cooperations among users with different levels of expertise, possibly at different sites and almost in real-time, in the development of the conceptualization of a problem domain or of the visualization of an algorithm.

Our paper begins with an overview of the FLAIR repository framework. Working examples of repository modules on search and automated reasoning are then presented. Classroom use of the repository is discussed. Finally, extensions of the system and its future as a learning environment that supports sharing and cooperation while allowing diversity will be addressed.

Initial experience in the use of the repository in teaching the introductory AI course at our institutions is taking place in the 1993/94 academic year.

2 FLAIR Repository Framework

The FLAIR repository is built upon an underlying system of objects, called Repository Elements (REs). Examples of REs are concepts, such as Heuristic Search; displays, such as the graphic representation of a search tree; and algorithms, such as the A* algorithm. Each RE is (optionally) persistent and is identified by both a system-generated ID (unique across all individuals and sites using a FLAIR Repository) and one or more human-generated names (homonyms and synonyms are allowed). This approach allows all REs to be shared between individuals and repository sites, through system-provided import and export facilities. Thus, new material developed at one site, covering a domain such as neural networks, can be easily incorporated into any other FLAIR repository. In addition, both instructors and students have the ability to maintain multiple versions of material on the same domain.

When users initially run the system, they are assigned to a variety of user-types (student, instructor, system developer, *etc.*). A user-type determines the visibility of REs to users. This can be customized by the instructor at the level of RE class (*e.g.*, a student user-type can/cannot 'see' an RE-Editor) or at the level of particular attributes and relations (*e.g.*, the student user-type can/cannot see the identification attribute of an RE).

The AI domain concepts can be viewed by the student either in tabular format or modeled as a conceptual graph. Concepts are represented as system objects that have both attributes (such as a text description) and relations to other concepts (*e.g.* search *is-part* of AI, heuristic search *uses* a heuristic function, *etc.*). Text presented to the student includes active text regions (hypertext), which enable the student to call up explanations of any unknown terms. The use of such hypertext combined with conceptual graphs for teaching has begun to prove itself in a variety of hypermedia-based instructional systems (Jonassen & Mandl 1990).

Algorithms are presented in active, graphical form animations, termed FLAIR Modules, with which the students can fully interact. Students have control over the algorithm's initial data and parameters, and can pause, single-step and change display speed at any point in the animation. Multiple views of the algorithm, ranging from its high-level aspects to detailed displays of the data structures it manipulates are available within the module. Recent studies have begun to demonstrate the impact that such a graphical approach to teaching systems can have on both student enthusiasm and learning (Badre *et al*, 1992; Brown *et al*, 1992). The FLAIR system enables the user to run multiple copies of the same or distinct modules with all copies running concurrently.

Central to the understanding of any algorithm is the ability to easily visualize and work with the data inputs and outputs of that algorithm. Editors (graphical wherever possible) are provided for each type of module data so that the student (and the instructor) can easily modify existing sample data or create new data sets for experimentation. Data can often exist in a variety of formats; given basic conversion routines, the system will automatically and transparently handle conversions. FLAIR also facilitates the use, as input data, of data produced as output by another module, enabling both the chaining of modules as well as the subroutine-like 'calling' of one module by another.

As mentioned previously, an emphasis in the design and implementation of the FLAIR system has been the ease with which instructors will be able to modify and add repository components. Development has stressed the production of system tools, extensible object-oriented libraries, and strong programming frameworks. REs may be added or modified by means of a tool, the FLAIR-RE editor. This editor consists of a basic text editor with added functionality allowing the definition of active-text regions and the specification of RE attributes and relations. Once an RE has been created, its relationships to other existing REs can be established graphically with the Navigator-Editor. Extensible object libraries exist that handle animation control, module data input and output, and that provide animation toolkits for dynamically displaying tree and list data structures. Together, these tools allow faculty (and students) to alter and extend textual explanations for concepts and other system components, to alter and extend existing conceptual graphs, and to create alternative or new conceptual graphs. This type of tool-based approach has also been suggested in (Murray and Woolf, 1992).

Developing new algorithm-demonstration modules at this time requires a considerable amount of knowledge of the system's programming language (Common Lisp) and graphical user interface development system (see section 4). Module developers will be aided by the availability of a Developer's Manual, which details the steps required to produce a new module.

3 FLAIR Repository Features

3.1 Navigator

Access to the conceptual network of the FLAIR system is provided through a graphical tool termed the Navigator (Fig. 1). The Navigator displays the REs of the system as nodes of a graph, connected by edges representing the relationships between those REs. Both the graph nodes and edges have different appearances (in color, shape and design) to differentiate the various RE classes and relationships. The Navigator provides the student with a graphical overview of the domain covered by the repository.

The RE classes and relation types currently displayed are controlled by two button panels (or palettes) which enable each type of RE class or relationship to be toggled on or off. Additional functions are available to the user by clicking with the mouse on any node. These functions allow users to delete a node, add new nodes and edges, inspect the node's contents, or mark the node. Marking a node is equivalent to a graphical 'bookmark' and allows the users to quickly return to that node if they later scroll the Navigator view away from that node into another part of the graph.

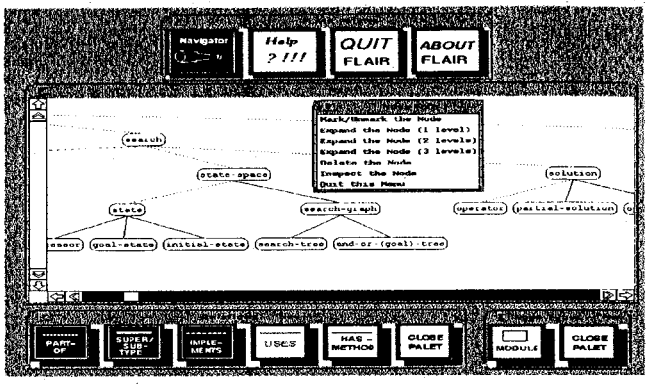

Figure 1. FLAIR Navigator

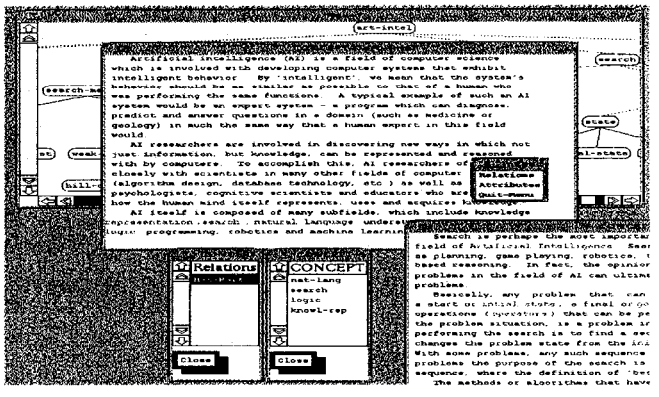

Figure 2. Inspector Windows

3.2 Inspectors

If the user chooses to inspect a node (RE) from the Navigator, a tool termed an RE Inspector is activated (Fig. 2). The Inspector displays an active-region-containing piece of text (hypertext) which describes that RE. Clicking on any highlighted (active) text term in the Inspector generates a new copy of an Inspector displaying the RE associated with that term. This enables the student to browse freely (non-sequentially) through the network of conceptual material in the repository. By clicking on any non-active region of the system, students can call up a set of menu-based browsers that enable them to view the attributes and relations of that RE. This presents the students with an alternative, table-based method for viewing the contents of any RE.

3.3 Modules

From an Inspector which is being used to view a Module RE, the students have the option of running the module. The students are initially presented with the choice of using the default data for that Module, selecting alternative existing data from the repository, or calling up an Editor tool to update an existing data set (or to create their own). Once the data set has been selected, the system displays the module's main window.

Two examples of modules are shown in Figures 3 and 4 – the Map-Search module and the Resolution and Unification modules. The Map-Search module is designed to demonstrate a wide variety of AI search algorithms within the context of searching a map for a path (or the shortest path) between cities on a map. [2] The Resolution and the Unification modules are designed to allow the student to experiment, respectively, with the resolution rule, and the unification algorithm. Fig. 3 shows a screen in which two copies of the Map-Search module are running, allowing the student to simultaneously view and compare the performance of two different search algorithms operating on the same data. Fig. 4 shows the Unification module running as a 'subroutine' of the Resolution module, *i.e.*, invoked to animate the computation of the unifier used by the resolution rule; the student has also selected (mouse clicked on) a piece of active text on the Unification module's main window, causing the display of the text describing the unification concept.

Each module's main display window is highly individualized to the particular aspects of the algorithm(s) that it aims to demonstrate. However, two sets of controls that are common to all modules can be seen in both Figs. 3 and 4 – the /One-Step/Pause/Proceed/Reset button set that serves as a common way of controlling module execution, and the Speed trill box, which controls the rate at which steps are executed. Other controls are particular to

[2] Additional details about the workings of an older version of the search module are in (Webster, 1991).

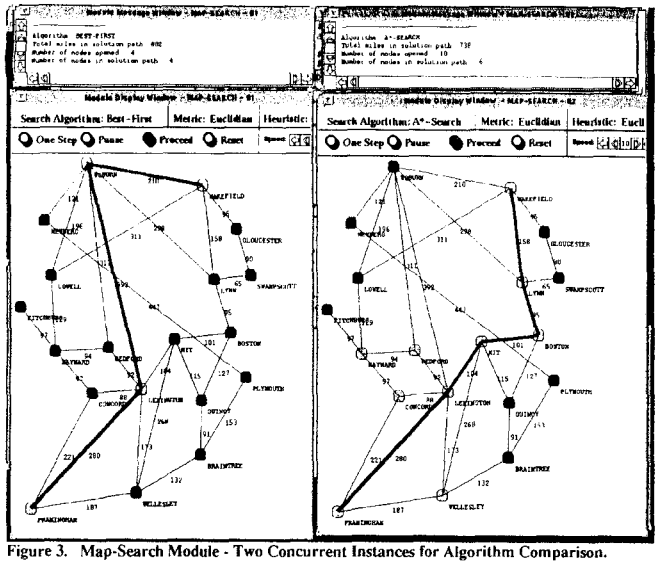

Figure 3. Map-Search Module - Two Concurrent Instances for Algorithm Comparison.

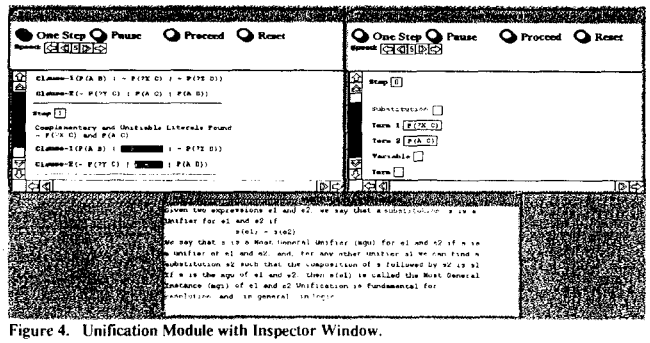

Figure 4. Unification Module with Inspector Window.

the type of algorithm upon which the module is designed – such as the buttons for setting the metric and heuristic on the Map-Search module's display window.

4 Platform and Portability

The FLAIR system is currently being developed and tested with students on SUN SPARC workstations. The system is written in Common Lisp (CL) with CLOS (Common Lisp Object System). The graphical user interface for FLAIR was developed using the Garnet system, a public domain X-Windows and lisp-based GUI development system developed at Carnegie Mellon University (Myers *et al*, 1992). The persistence facilities of FLAIR are implemented in CL using standard file system support, not layered on a database management system.

The FLAIR system is portable to any hardware/software environment that supports a CL that runs Garnet.

5 Pedagogic Use

The FLAIR system is being used this academic year (1993/94) within the laboratory component of undergraduate AI courses at Temple, Millersville, and Villanova Universities. The FLAIR system is available to the students in their laboratory sessions, along with the programming languages Lisp and Prolog. Remote access using the X-Windows protocol will be available to Drexel University students in the undergraduate AI course.

The current contents of the FLAIR system cover some aspects of the AI domains of search and automated reasoning. Students can freely explore and utilize the conceptual material in FLAIR to supplement their classroom instruction on these subjects.

Three closed-lab projects using FLAIR were undertaken in the undergraduate AI course at Temple University during the Fall 1993 Term. The first project had students use the map-search editor to design a city map and a search problem (designating start and goal cities) for which the A* algorithm finds a shorter path than does the Best-First algorithm. Students were then expected to explain the differences between these algorithms with reference to their search problems. Students enjoyed the module's animated environment, but, more importantly, the project revealed that most students were unable to predict or explain how the algorithms would perform on specific kinds of problems even though the algorithms had been covered in both lecture and text. The map environment encouraged students to explore how problem characteristics affected the performance of these algorithms, and also provided means for explaining these differences by allowing students to display and examine the pseudocode and lisp code for each algorithm.

For the second project, students were asked to translate an English description of a problem into a set of predicate logic formulas, and to use the clausification and resolution proof procedure modules available in FLAIR to solve the problem. This project allowed students to explore the interdependencies of the clausification, unification, and resolution algorithms, as the modules notify users where these interdependencies occur and permit the user to run the algorithm animations concurrently.

A primary objective of the third project was to help students develop a better understanding of the conceptual structure of a domain than is normally possible with the discursive style of a term paper. Students formed small teams for this project. The task was to develop a concept network for fuzzy logic, a subdomain of AI which had not yet been included in the Navigator and browser. In addition to defining the structure of this subdomain, students were asked to include a small set of concepts from areas other than fuzzy logic which would function as bridge concepts between fuzzy logic and other AI topic areas. FLAIR allows users to extend the Navigator concept map with segments which are stored in the user's private area but integrated into the common concept map when the user activates a FLAIR image. Students may thus customize FLAIR's concept map, and compare different versions. Competition among teams on this project motivated students to go beyond the textbook and lecture

explanations of the subject to research the field.

These preliminary experiences with FLAIR in the lab were, as expected, interfered with by a number of program bugs and system problems. Yet, these experiences suggest how FLAIR could be used to enhance lecture and text through laboratory projects in ways not typically available. In the first two projects, students benefitted from an environment that supports exploration and that could be used to supplement theoretical presentations with concrete experimentation. In the third project, FLAIR provided an environment for an unusual kind of project that directly extends the theoretical component of the course through laboratory activity.

6 Future Extensions

Planned additions to the system include:

1. **Demo Editor** – this tool will enable instructors to quickly produce module demos: pre-recorded sessions with a particular module using a particular set of input data and accompanied by a display of explanatory text.

2. **Student Notebook** – a tool which will allow the student to save the current initial conditions for a module to a private 'notebook,' with the student's textual annotation explaining why this particular module 'run' is important. Such a facility will be used in the development of student projects.

3. **Intelligent Retrieval** – a tool enabling a user to enter a query containing textwords and retrieve any RE whose associated textual description matches that expression.

4. **Client-Server Architecture** – a facility, transparent to the user, that will allow access to multiple file-servers (at the same physical site or across institutions) as if they formed a single FLAIR repository. This facility will be based on the WAIS/Gopher model (Krol, 1992) and will greatly increase the ability of instructors to share FLAIR material, both intra- and inter-site.

7 Conclusions

The FLAIR system is a repository designed to augment instruction in AI by providing faculty with a variety of hypertext-based conceptual materials and highly interactive, graphical demonstration modules to illustrate AI techniques and algorithms. Included within the system are a variety of tools, software libraries and programming frameworks to allow both instructors and students to easily modify and extend the basic system. The benefits of these libraries and tools include: i) easy customization of modules and demos; ii) pooling of resources across courses and institutions; and, iii) support for student programming projects.

The distributed, object-oriented architecture developed for the repository supports the sharing of both conceptual material and modules across sites using the system. It is our intention and hope that FLAIR will prove to be more than an immediately-useful 'off-the-shelf' system for teaching introductory AI. We see FLAIR as a tool that will enable instructors in AI and other subjects to pool their resources across institutions, magnifying the gain each derives from the time and effort spent extending the system.

A preliminary version of the FLAIR software and material can be obtained by contacting the authors.

BIBLIOGRAPHY

1. Aiken, R., Ingargiola G., Hoskin N., Solley J., Wilson J., Christensen M., Papalaskari M.A. (1992) "Providing laboratory support for the introductory AI course," ASEE-IEEE, Frontiers in Education Conference, Tennesee, 714-718.

2. Aiken, R., Allemang, D., Wehrle,T. (1992) "Designing laboratory modules for novices in an undergraduate AI course," ACM SIGCSE Bulletin, 24 (1), 76-80.

3. Badre, A., Beranek, M., Morris, J.M., Stasko, J. (1992) "Assessing program visualization systems as instructional aids." Proc. 4th Intl. Conf. Computer-Assisted Learning, Wolfville, Canada, 87-99.

4. Brown, C., Fell, H., Proulx, V.K., Rasala, R. (1992) "Programming by experimentation and example." Proc. 4th Intl. Conf. Computer-Assisted Learning, Wolfville, Canada, 136-147.

5. Jonassen D., Mandl H., Eds. (1990) "Designing Hypermedia for Learning," Springer-Verlag.

6. Krol, Ed (1992) "The Whole Internet Users Guide and Catalog." O'Reilly and Associates Publisher.

7. Murray, T. and Woolf B.P. (1992) "Tools for teacher participation in ITS design." Proc. 2nd Int. Conf. ITS, Montreal, Canada, 593-600.

8. Myers B.A. *et al* (1990) "Garnet: Comprehensive support for graphical highly interactive user interfaces," 23(11), 71-85.

9. Webster R., Ross P. (1991) "Useful artificial intelligence tools – A review of heuristic search methods," IEEE Potentials Journal, 10(3), 51-54.

10. Webster R., Ross P. (1992) "A workstation laboratory to improve undergraduate instruction in Artificial Intelligence," Proc. of the Annual American Society for Engineering Education (ASEE) Conf., Toledo, Ohio, June 21-25, 1992, 448-455.