AC 2008-1545: INTELLIGENT TUTOR FOR LADDER LOGIC PROGRAMMING

Sheng-Jen Hsieh, Texas A&M University
Dr. Sheng-Jen (“Tony”) Hsieh is an Associate Professor in the College of Engineering at Texas A&M University. He holds a joint appointment with the Department of Engineering Technology and the Department of Mechanical Engineering. His research interests include engineering education, cognitive task analysis, automation, robotics and control, intelligent manufacturing system design, and micro/nano manufacturing. He is also the Director of the Rockwell Automation laboratory at Texas A&M University, a state-of-the-art facility for education and research in the areas of automation, control, and automated system integration.

Yuan-Teng Cheng, Texas A&M University
Yuan-Teng ("Randy") Cheng graduated from Texas A&M University with MS in Computer Science in 2007.
Intelligent Tutor for Ladder Logic Programming

Programmable logic controllers (PLC) are used for many industrial process control applications. Learning to write ladder logic programs for PLC control is an important and challenging task. However, the learning of ladder logic is often hindered by limited PLC availability due to expensive lab setup, limited lab time, and high student-instructor ratios. As part of a larger effort to develop a web-based learning system called Virtual PLC, an intelligent tutoring system (ITS) has been designed to teach students how to write and debug ladder logic programs. This paper describes procedures employed in developing the ITS component, evaluation of system performance, subsequent enhancements to the system, and future research directions.

Background

The programmable logic controller (PLC) has been described as one of the most ingenious devices ever invented to advance the field of manufacturing automation\(^1\). Thousands of PLCs have been used for such applications as monitoring security, managing energy consumption, and controlling machines and automatic production lines. As an essential part of manufacturing automation, PLCs are covered in many automation and control-related courses, such as Computer-Aided Manufacturing, Computer-Integrated Manufacturing, Industrial Control, and Manufacturing Automation and Robotics. Students from other disciplines are also exposed to PLC technology. Research by Frost & Sullivan indicates that the world market for programmable logic controller will continue to grow as units become smaller, more functional, and more able to work in tough environments\(^2\). The total PLC and software market was $7.2 billion in 2006 with steady growth expected over the next five years. There is a great need for engineers with strong skills and knowledge in this area.

An important component of learning is practice. This is especially important when learning programming. However learning PLC programming is different from learning general purpose programming because equipment interfacing and control is involved.

Unfortunately, educational institutions often lack the resources needed to help students to become proficient with PLC programming. For example, a Manufacturing Automation and Robotics course typically uses programmable logic controllers, sensors, robots, and machine vision systems in teaching automated manufacturing system design. However, students’ learning is hindered by obstacles such as:

- High faculty-to-student ratios: For example, the ratio for Manufacturing Automation and Robotics at one major university is 1:36 for lectures and 1:18 for each of two lab sections.
- Limited lab access: Students may only use equipment during scheduled lab times.
- Limited resources to support students outside labs and the classroom: In many cases, no lab assistant support is available.
- Limited equipment to support lab assignments: Because lab equipment is often expensive, students must often work in groups. For example, an industrial scale PLC—such as an Allen Bradley RSLogix 5550 processor and a set of I/O cards—costs about $8000.
To help offset these obstacles, in recent years, a variety of computer and networking technologies have been developed. For example, remote laboratories (also known as e-Lab, Tele-Lab)\textsuperscript{3,4}, virtual laboratories\textsuperscript{6,7,8,9}, and hybrid laboratories\textsuperscript{4} have been developed to reduce lab equipment setup costs and increase accessibility. Other developments include the use of interactive\textsuperscript{7}, multimedia-enhanced\textsuperscript{10,11}, and integrated\textsuperscript{12} approaches and the Design-Build-Test concept\textsuperscript{13} to make learning more interactive and visual. However, there have been relatively few attempts to use technology to teach PLC programming.

One of the authors was recently awarded a National Science Foundation grant to develop an Integrated Virtual Learning System (IVLS) for PLC education that incorporates intelligent tutoring systems, simulations, and animations. A prototype version of this system, known as Virtual PLC, can be found at \texttt{http://etidweb.tamu.edu/hsieh/Hsieh_VirtualPLC.html}. Previously reported work includes the design, development, and successful evaluation of Virtual PLC courseware on the topic of PLC timer instructions\textsuperscript{14,15} and a ladder logic toolkit\textsuperscript{16}. This courseware was found to be instructionally effective, and students’ subjective impressions of the system were positive overall. However, the focus of these lessons was on the usage and syntax of PLC instructions. The system described here was designed to teach PLC programming and debugging. It does this by helping students to visualize how an automated system should work and to understand how to control a real-life or close to real-life system using ladder logic. It also allows students to manipulate an animated model of the system in order to explore what happens under different conditions.

The design task was approached in two phases. First, we established the feasibility of developing a web-based ladder logic programming and debugging environment using a database of problems and instructional information. Second, we began to make the environment adaptive. The programming problems used were designed to address common programming mistakes and misconceptions. The two phases are described below.

**Phase I: Feasibility study**

A web-based PLC debugging tutor was developed to help students learn to debug PLC programs. The tutor includes ten problems designed to test learners’ understanding of ladder logic diagrams. Learners write PLC programs to solve these problems, and then use a debugging tool to edit and verify their programs. When the program is executed, an animation shows the action of devices, such as buttons, switches, fans, and lights. The system also checks for errors and directs learners to debug their program.

**System Structure**

The system structure is depicted in Figure 1 below. A learner logs in and chooses one of the ten problems available. The system then presents the problem goal, required background knowledge and overview, instructions about how to use the interface, and a problem statement (Figure 2). The learner then advances to the Configuration Selection page (Figure 3) and selects a rung template for use in programming. This page helps test whether the learner has the necessary background knowledge to complete the problem. After the learner selects an appropriate template, he/she is allowed to advance to the program development page (Figure 4). The
program development page interface allows learners to drag ladder logic symbols from a toolbox onto the rung template. The learner can then run the program and manipulate an animation of the results (Figure 5). For example, in Figure 5, learners can click the spring return buttons on the image to test their program. As they click the buttons, the status of the corresponding ladder logic components will change in real time. If the program was written correctly, lights L and M will turn on. Otherwise, one or more error messages will be displayed in the green Results area. Learners can go back to the program development screen to change their program based on the instructions in the error messages.

Figure 1. Phase I System structure.
Problem Statement:
A milling machine (M) and its lubrication (L) both have start/stop control systems. L must be running before M can be started. Furthermore, if L stops, M must also stop. Normally Open Spring Return (N.O.S.R.) Start and Stop buttons are used to control the machine.

Instructions:
In order to perform the described task, click on the appropriate ladder diagram located to the right.
System Implementation

To implement the tutor, we used Apache Server, PHP 5 programming language and Microsoft Access 97 database (Figure 6). Once the learner enters the program development page, completes some drag-and-drop actions, and then runs the program, the system sends a command to the Apache Server. Then, the PHP scripts behind the server executes the command to store/retrieve data from the Access 97 database. On the client side, the tutor is programmed in Flash.
Evaluation

The ladder logic debugging tool was evaluated by 25 undergraduate students enrolled in a manufacturing automation and robotics course. Students completed a pre-test before using the prototype and a post-test afterwards. Differences between students' pre- and post-test scores were significant ($\alpha = .05$). In addition, students responded positively to a survey that asked their opinions about the system's effectiveness, ease-of-use, and relevance to their education. Students also completed a learning styles survey (Felder and Soloman's Index of Learning Styles). It appears that the debugging tool's visual, exploratory is a good fit for students who consider themselves to be active, visual learners.

Figure 7 summarizes the means for the opinion survey for the ladder logic-debugging tool. Student ratings were positive for all items. In general, students felt that the tool was interactive, relevant, and easy to use and understand. Figure 8 shows a summary of results for all four attribute continuums from the Index of Learning Styles for the students in this study. These data suggest that the students tended to be more visual than verbal and more active than reflective as learners.

Figure 7. Opinion Survey Means for Ladder Logic Debugging Tool.
Phase II: Adaptive Environment Design

Based on the results of the Phase I evaluation, we have made the following improvements and enhancements:

1. Added a pre-test that learners can complete before beginning problem-solving. The goal of the pre-test is to identify which problems are most suitable for the learner’s current knowledge about this PLC domain. The pre-test will only be administered when learners first begin using the system.
2. Created a Generic Toolbox to extend the predefined Rung configuration and component-limited toolbox.
3. Allow the learner to change Logic and Examine Bits in whatever location desired.
4. When a rung program has a bug, allow learners to make minor modifications to the rung configuration and re-run the program immediately.
5. Modified errors in Error Message Debugging Text.
8. Designed a smart Heuristic Function Algorithm to map and find out user errors.
9. Created rules in PHP to identify which concepts a user doesn’t understand.
10. Extracted source code written in Flash ActionScript and re-implement using PHP code to improve system performance.
11. Focus on one error at one time in guiding users through their errors.
12. Added more detail to user action tracking/ time-recording functions.

Figure 9 shows the structure of the adaptive version of the debugging tutor. This version was developed with the following design goals in mind.
a. Pretest questions should be individualized. Instead of assuming that the learner doesn’t know any PLC programming concepts, we want to offer a dynamic pre-test that can classify learners according to their level of knowledge, such as beginner, intermediate, or advanced. The pre-test questions are generated one at a time, and the content of the next question is based on the learner’s performance on previous questions. Learners are tested starting from basic concepts and the difficulty level is gradually adjusted based on what the learner knows. This allows the pre-testing process to be more efficient.

b. Add a student model. The system includes an Error Patterns database that records patterns of errors in rung programming. If a learner attempts to run a program that contains an error, the system will use built-in heuristic functions to identify the error pattern in the database that most closely matches the learner’s error. Once the most similar error pattern has been identified, the misunderstood concept(s) will be displayed for the learner to see.

Figure 9. Phase II System Structure
c. Add a tutor model that can address learner misconceptions. The student model detects any misconceptions the learner may have and informs the learner. The learner can then choose to complete a tutorial designed to address the misconception. The Tutor model will bring up the tutorials, present the concept(s), and then generate some test questions to evaluate the understanding of the learner. Once the learner has mastered the concept, he or she will be given the opportunity to try to solve the original problem again.

d. Continuously improve the system’s ability to diagnose misconceptions. Whenever the learner runs a program that contains an error, the system will add the incorrect configuration to the Error Patterns database. The system will also maintain a record of the tutorials the learner chooses to complete in the course of solving each problem. Thus the Error Patterns database will gradually accumulate more error patterns and be better able to diagnose learners’ errors in the future.

System Implementation

We have developed the heuristic function described in the design goal (b) above. The purpose of the heuristic function is to map learner’s incorrect rung patterns to existing patterns in the error pattern database. Based on the programming logic controller application, our cumulative heuristic function $H(n)$ is listed as equation (5) below.

\[
\begin{align*}
    h_1(n) &= \text{Redundant Component} : -1 \\
    h_2(n) &= \text{Rung Length Difference} : -|\text{Rung Length Difference}| \\
    h_3(n) &= \text{Matching Component} : +1 \\
    h_4(n) &= \text{Misplaced Component} : \\
    &\quad \sum_{i=1}^{\text{length(Rung)}} \text{Position Diff of Component}_i \\
    H(n) &= \sum_{i=1}^{4} h_i(n)
\end{align*}
\]

Following is an example of how our heuristic function works. Figure 10 shows the Rung configurations in PLC and mapping sequences, or tuples, in the database, and Figure 11 shows the utility calculation of our heuristic function. From the result, we can see that the current learner’s error rung configuration maps to Rung 3 in the database.
Rung Configuration in PLC     Mapping Sequence in the Database

Rung 1: |------]/[-----][-----( )-----|  {<XIO,1,0>, <XIC,2,0>,<OUT,2,1>,<OUT,3,0>}
        --( )--

Rung 2: |-----]/[-----][-----[-----[-----( )-----|  {<XIO,1,0>, <XIC,2,0>,<XIC,3,0>,<XIO,3,1>,
        --]/[--<OUT,4,0>}

Rung 3: |-----]/[-----( )-----][-----|  {<XIO,1,0>, <OUT,2,0>,<XIC,3,0>}

Rung 4: |-----]/[-----( )-----|  {<XIO,1,0>, <XIC,1,1>,<OUT,2,0>}
        --]/[--

Current Learner’s error rung configuration:
Rung : |----- ( )-----][-----]/[-----|  {<OUT,1,0>, <XIC,2,0>,<XIO,3,0>}

Figure 10. Example of Rung Configurations

<table>
<thead>
<tr>
<th>Rung #</th>
<th>Redundant Component</th>
<th>Length Difference</th>
<th>Matching Component</th>
<th>Misplaced Component</th>
<th>H(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-3-3=0</td>
<td>+3</td>
<td>(1-3)+(2-2)+(3-1)</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
<td>-3-4=-1</td>
<td>+3</td>
<td>(1-4)+(2-2)+(3-1)</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-3-3=0</td>
<td>+3</td>
<td>(1-2)+(2-1)+(3-1)</td>
<td>5*</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-3-2=-1</td>
<td>+3</td>
<td>(1-2)+(2-1)+(3-1)</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 11. Calculation of Heuristic Function H(n)

As in Phase I, we used Apache Server, PHP 5 programming language and Microsoft Access 97 database (Figure 6). After the learner enters the program development page, completes some drag-and-drop actions, and runs the program, the system sends a command to the Apache Server. The PHP scripts behind the server will then execute the command to store/retrieve data from the Access 97 database. Figures 12 to 18 show screen shots from the system.
**Problem Statement:**
A fan can be turned On and Off by using a normally open switch (connected to input #1) and a normally closed spring return button (connected to input #3). One is located at the front door; the other one is located at the back door. The fan, which will be activated if either one button is pressed, is connected to output #2 in the Relay Output module at slot 9.

**Instructions:**
Click on either the parallel or series ladder diagram.

**Results:**

---

**Figure 12. Sample problem statement and instructions.**

---

**Figure 13. Sample problem-solving page.**
Figure 14. Select Parallel/Serial position.

Figure 15. Select TON logic

Figure 16. Select Examine Timer Bits.
System Evaluation

System evaluation has so far focused on four different areas—learning algorithm robustness, system adaptivity, student evaluation, and large scale population simulation model. A few learning algorithms were compared to examine the mapping accuracy of user error patterns with existing patterns in the database; the proposed learning algorithm was most accurate. The purpose of the adaptivity testing was to evaluate if the system’s accuracy in mapping error patterns would increase if student sample size increases; results suggested this was the case. In the student evaluation, we found that before using the tutoring system, participants had knowledge proficiency levels between 0.5 and 0.7. After using the ITS, they learned all the required concepts within 2 to 3 hours. A simulation model was developed to mimic a 1000
student population with different levels of knowledge proficiency. Results suggest that a user will require less time to learn if he/she has higher initial knowledge proficiency.

Future Directions

Future directions include constructing 1) a knowledge architecture based on system schematics; 2) learning algorithms that takes a user’s psychological and physical condition into consideration in developing a student model, 3) adaptive procedures that self-improve based on the number of student samples, and 4) an interactive graphical user interface design that includes with voice, visual, and other sensory devices.

Acknowledgments

This material was supported by a National Science Foundation Course, Curriculum, and Laboratory Improvement (CCLI) grant (No. 0341287) and a gift from Rockwell Automation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation or Rockwell Automation.

Bibliography