Playing to Program: An Intelligent Programming Tutor for RUR-PLE

Marie desJardins and an Ensemble of Students
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore MD 21250
mariedj@cs.umbc.edu

Abstract

Intelligent tutoring systems (ITSs) have proven their effectiveness in contributing to student learning. ITSs are automated programs that provide students with a one-on-one tutor, allowing them to work at their own pace, so they can spend more time on their weaker areas of the subject matter. RUR–Python Learning Environment (RUR-PLE), a virtual environment to help students learn to program in Python, provides an interface for students to write their own Python code and then be presented with a visualization of that same code [CITE]. The RUR-PLE system provides a sequence of learning lessons for students to explore. We have extended RUR-PLE to provide an intelligent tutoting system interface that consists of three components: (1) a student model that tracks student understanding, (2) a diagnosis module that provides tailored feedback to students, and (3) a problem selection module that guides the student's learning process. In this paper, we describe the basis RUR-PLE system and our extensions, and present the results of a user study in which we evaluated the effectiveness of our three ITS modules.

1. Introduction and Motivation

ADD introduction and motivation.

2. Related Work

ADD: A discussion of general work on ITS, programming tutors, CS1/iteration education research, etc.

3. Background: RUR-PLE

ADD general overview of RUR-PLE and the functionality it provides.

4. Infrastructure

ADD description of concept map, how we built/tested it (i.e., justification for these concepts and why they're connected as they are, and how we instantiate and track it for a specific user (presumably using some kind of Bayesian updating). Idea: validate/finalize it by some testing process on a group of students (i.e., is it in fact the case that students in general need to understand concept X before they can apply concept Y) – use a problem suite (where each problem includes known concepts) to measure these dependencies.

5. Pre-test

ADD: Discussion of diagnosis module: our state-based approach to comparing student solutions to one or more model solutions per problem. Where do the model solutions come from? What does the comparison consist of? How is the resulting "diff" used to generate suggestions (and update the student model)?

6. Problem Selection

ADD: How is the student model used to generate and select problems for the student to work on? What is the motivation for our approach, and how does it work?

7. Experimental Design

ADD: Methodology: design of the user study (set up in such a

8. Resultes

ADD: Resultes

9. Conclusions and Future Work

ADD: What have we contributed? What are the takeaway lessons? What would we work on next?

References