On Improving the Effectiveness of Open Learning Environments Through Tailored Support for Exploration

Andrea Bunt[1], Cristina Conati, Michael Huggett & Kasia Muldner
{*bunt, conati, mikey, kmuldner*}*@cs.ubc.ca*

Department of Computer Science
University of British Columbia
2366 Main Mall
Vancouver, B.C. Canada V6T 1Z4
Phone: (604) 822-4632
Fax: (604) 822-5485

Abstract Open learning environments can be beneficial for learning in ways not available in more tutor-controlled systems, because of the active role the learner plays in knowledge acquisition. However, it has been shown that not all learners are proficient in unconstrained exploration, restricting their ability to learn effectively in these environments. In this paper we present the Adaptive Coach for Exploration (ACE), a prototype computational framework that supports active exploration in an open learning environment by providing tailored support to overcome specific student difficulties.

ACE provides students with a highly-graphical, exploratory learning environment in the domain of mathematical functions. A Student Model assesses student knowledge and exploratory behaviour using a Bayesian network; ACE's Coach uses this assessment to generate tailored hints that support the exploratory behaviour of those students who would otherwise have trouble learning in an unsupervised environment.

After describing ACE's components, we present the promising results of a preliminary user study that gauges the system's effectiveness.

1. Introduction

Open learning environments have been the subject of extensive research in the field of computer-supported learning. These environments, also known as exploratory or discovery environments, place less emphasis on explicit instruction and more on providing the learner with tools that support learning through unconstrained exploration of the target instructional domain [3, 18, 19]. Advocates of open learning environments believe that, through active involvement in the knowledge acquisition process, the student can gain a deeper and more structured understanding of the domain. At the other end of the spectrum, supporters of more guided ways of learning argue for the effectiveness of tutor-controlled environments that monitor and structure the learning process through focused activities [2, 12, 16].

[1] Research funded by NSERC PGS A

While there is substantial evidence on the effectiveness of environments that rely on some degree of tutor control [1, 5, 6], user evaluations of pure exploratory/discovery environments have produced mixed results. In particular, successful learning in these environments seems to depend strongly on the student's learning style [10, 11] and on meta-cognitive skills which contribute to effective exploration, such as the ability to formulate hypotheses, perform experiments, draw conclusions based on the results, and monitor one's progress in the learning process [4].These results suggest that the effectiveness of open learning environments can be improved by providing additional support to the learning process. Two approaches have been followed in this direction.

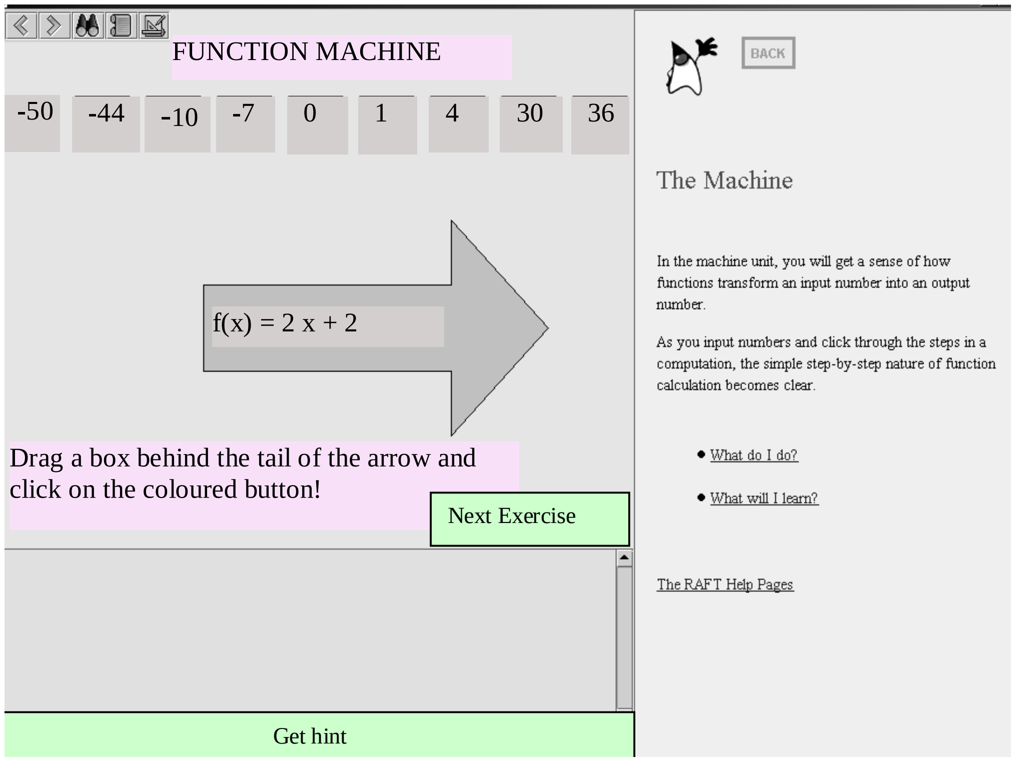
The first approach is to provide supplementary cognitive tools specifically designed to scaffold the application of the relevant meta-cognitive skills [7, 14, 18]. However, the results obtained with this approach indicate that even when very carefully designed, cognitive tools can sometimes interfere with the learning process. This is especially true if their use is imposed on all learners, even the ones who do not need the extra scaffolding because they already possess and apply the relevant meta-cognitive skills.

The second approach is to provide the students with more active and explicit instruction, tailored to their specific difficulties in the exploratory process. This approach is quite difficult, since it relies on the capability to monitor and understand the student's unconstrained exploratory behaviour. Nonetheless, it has the advantage of providing help for those students who have problems learning through exploration, without affecting others who can explore effectively.

In this paper, we describe ACE (Adaptive Coach for Exploration), a prototype intelligent learning environment that follows the second approach by offering tailored support for the exploration of mathematical functions. ACE monitors the exploratory actions that students perform through its interface and tries to detect when a student is experiencing difficulties with the exploration process. When necessary, the environment generates interventions aimed at helping the student overcome these difficulties, while giving them a sense of control and freedom.

Because of the difficulty of monitoring a student's behaviour in an open learning environment, only a few other environments have pursued similar approaches. In [17] the authors present a student model to assess and support the process of hypothesis testing. This model relies on the student being active enough to generate hypotheses and so does not assist the student in searching the space of possible hypotheses. Similarly, Smithtown [11], a discovery learning environment in the domain of microeconomics, helps students structure their experiments by guiding them through a fixed sequence of steps, but does not address the needs of less active students who have problems initiating experiments and making predictions. Belvedere [8], an environment that provides graphical tools to build scientific arguments, also helps students improve their arguments based on predefined syntactic and consistency relations among argument components. However, Belvedere does not actually parse the students' arguments nor does it monitor the data collection process that the students engage in prior to argument formation.

In ACE, we focus more on eliciting students' exploratory behaviour, by explicitly guiding more passive learners to use ACE's interface tools effectively in exploring the target domain of mathematical functions. In the following sections of this paper, we first describe these tools. Next, we introduce both the probabilistic student model that ACE uses to assess students' exploratory behaviour, and ACE's coaching component. Finally, we report on the results of a preliminary study that evaluates ACE's effectiveness in promoting learning by supporting exploration.



Drag a box behind the tail of the arrow and click on the coloured button!

The Machine

In the machine unit, you will get a sense of how functions transform an input number into an output number.

As you input numbers and click through the steps in a computation, the simple step-by-step nature of function calculation becomes clear.

Figure 1: Machine Unit

2. Description of the System

The ACE system consists of three components: a Graphical User Interface (GUI), a Student Model, and a Coach. The GUI presents interactive activities geared at stimulating exploration of mathematical functions. The Student Model interprets interactions with the student in order to determine the effectiveness of the student's exploratory behaviour and level of knowledge. The Coach monitors the student's traversal of the curriculum, providing tailored situation-dependent suggestions and hints aimed at improving the student's exploration of the available material.

2.1 The GUI

The GUI is designed to allow the student to explore functions in as many ways as possible. Most of ACE's information is displayed in its main window (see Figure 1, left-top panel). The upper area of the window is a graphical display that can draw and animate text, pictures, and computable shapes such as function curves. This is the main area in which the student interacts with the system. Below this is a Feedback panel (Figure 1, left-bottom panel), in which hints from the Coach are printed. The right side of the window is in effect an HTML browser containing hyperlinks to the system's help pages, which include both interface instructions and domain-related help.

2.1.1 Three Unit Types

Currently, ACE presents the student with three different units – the Machine, the Switchboard, and the Plot unit, which are loosely based on the material in the pre-calculus

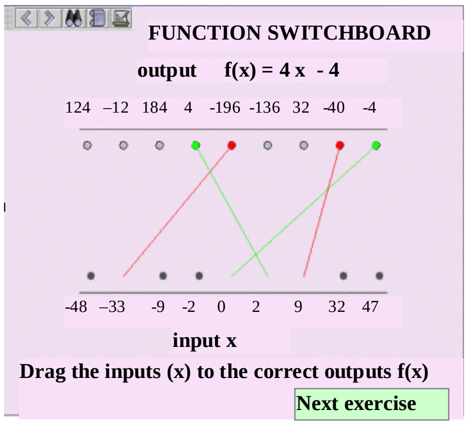


Figure 2: Switchboard Unit

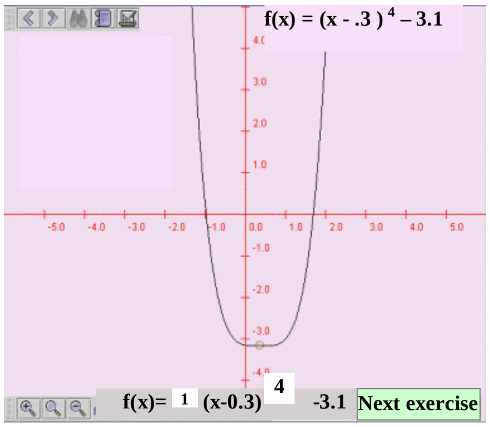


Figure 3: Plot Unit

section of [13]. Each unit contains a set of exercises; each exercise presents the student with a different function to explore. The units and exercises are initially shown in sequential order, providing increasing complexity of interaction as the student moves through the curriculum. The student can move to the next exercise by clicking on the “Next exercise” button. Also, the student can choose to move to any exercise by using the Lesson Browser tool (explained below) at any time.

The Machine unit (Figure 1) shows the student how a numeric input to a function is processed into an output. The student drags an input to the “machine”, which generates the output. The student is presented with a variety of inputs to explore. The student clicks through the ‘steps’ involved in the calculation, and can view the intermediate result at each step. Animation is used as an added stimulus – in order to draw the student’s attention to it, the output generated by the Machine appears in a pink circle after moving across the screen.

The Switchboard unit (Figure 2) requires more active thinking from the student, in that the student is given the opportunity to explore the mapping of a range of inputs onto a range of outputs. As the unit’s name might imply, the functionality is very much like a switchboard – each of the inputs has a ‘dragball’ next to it that the student can drag to any ‘socket’ next to an output number. As it is dragged, a line continuously connects it to the input. If the student succeeds in connecting the input to the correct output, the dragball and the connect line turn green, otherwise they turn red. The student may reconnect an input to a different output at any time.

The goal of the Plot unit (Figure 3) is to help the student explore the properties of graphs and equations, and relationships between the two. The interface contains both an equation box (found in the lower left corner of Figure 3), which shows the current function equation, and the corresponding graph of that function, displayed in an x-y plane. The student can manipulate the graph of the function either by dragging it around the plane or by typing directly into the equation box. Updating the graph automatically updates the function equation, and vice-versa.

2.1.2 Tools

Although the above units and corresponding exercises are initially presented in a predefined sequence, we want students to be able to freely explore the curriculum. Therefore, the GUI toolbar holds buttons to allow the student to move forward and backward through the curriculum, as well as a Lesson Browser (Figure 5). The Lesson Browser shows all exercises, and allows the student to go to any exercise by clicking on it. The toolbar also contains an

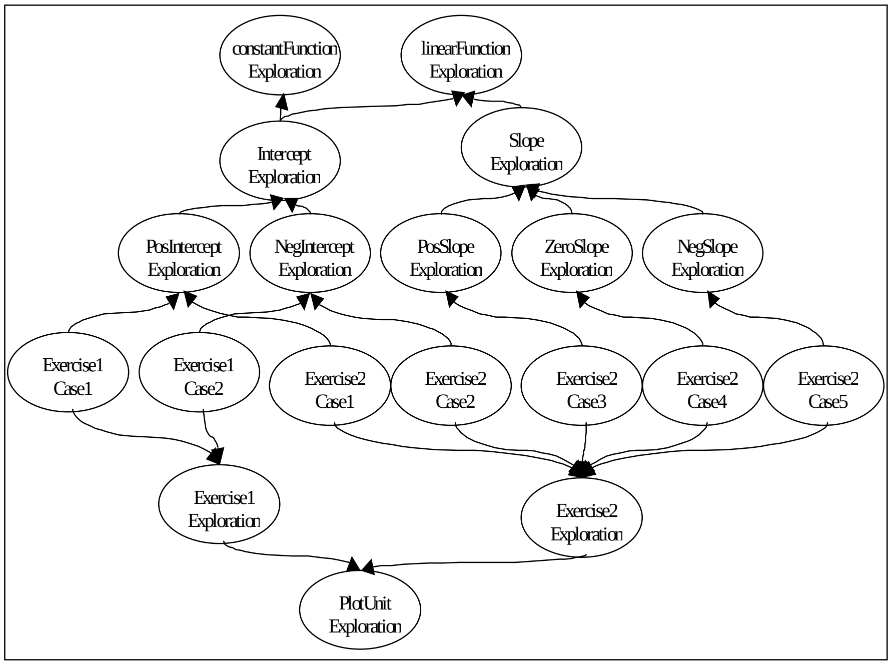


Figure 4: An example portion of the Bayesian Network for the plot unit.

Exploration Assistant, a tool to help the students organize their exploration process that will be described in more detail in a later section.

2.2 The Student Model

The Student Model monitors the student's interaction with the system in order to determine whether the student is effectively exploring the environment and gaining an understanding of the domain.

The Student Model must assess the student's behaviour with relatively sparse information. The model can view low-level information such as the inputs and outputs entered by the student, but it does not have any access to the student's underlying reasoning. While this restricts the level of assessment, it does allow for a more natural interaction with the system where the student is allowed to freely explore the environment without imposition. Given the limited information available to the model, assessing the student's behaviour involves a great deal of uncertainty, which we handle using the probabilistic reasoning framework of Bayesian networks [9].

The Student Model's Bayesian network consists of two types of nodes: one that assesses the effectiveness of the student's exploratory behaviour (exploratory nodes) and the other representing the student's understanding of the domain concepts. Exploratory nodes in the network represent exploratory behaviour at different levels of granularity (see Figure 4): the effectiveness of the student's overall exploration, the exploration of individual units, the exploration of individual exercises, and the exploration of concepts, such as 'slope' and 'intercept'. Concepts are modeled in the network in a hierarchical fashion. For example, the general concept of slope exploration consists of the more specific concepts – the exploration of positive slopes, negative slopes and the zero slope. The Conditional Probability Tables in

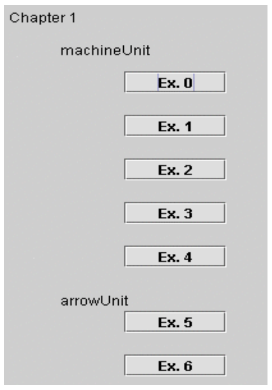


Figure 5: The Lesson Browser

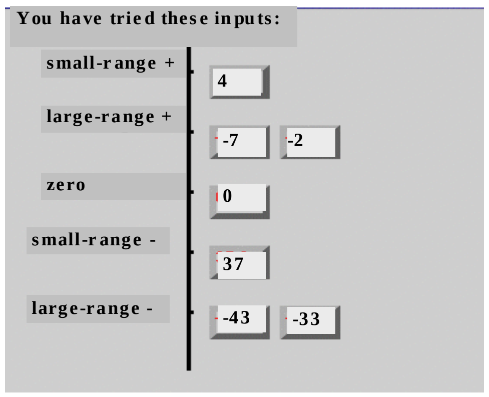


Figure 6: The Exploration Assistant

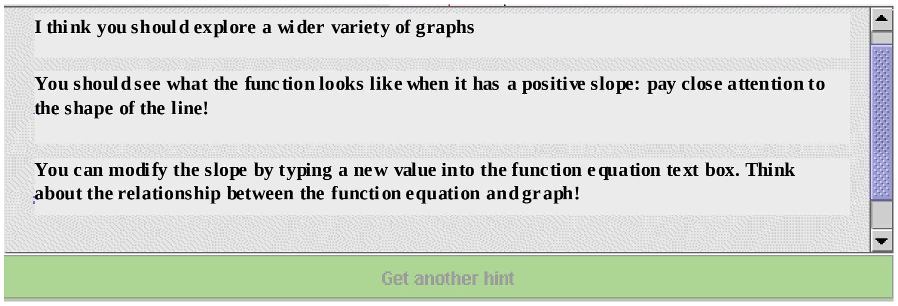
the network are constructed using our initial estimates; empirical evaluations will be needed to verify and refine these probabilities.

In order to assess the effectiveness of the student's exploratory behaviour, the Student Model looks for evidence that the student is exploring an exercise's salient concepts, referred to as *relevant exploration cases*. The relevant exploration cases for a particular exercise depend on both the current unit and current function being explored. For example, in the Machine and Switchboard units, the student should explore all of the different categories of inputs available, such as small positive inputs, large positive inputs, small negative inputs, large negative inputs, and zero. In the Plot unit, the student should explore how modifying each of the different components of the function (e.g. the slope coefficient) changes the shape of the graph, and vice-versa.

The Student Model uses evidence that each relevant exploration case has been sufficiently explored to both assess how well the student has explored a particular exercise, as well as how well the student has explored concepts that appear in multiple exercises. As the system does not know ahead of time the exact number and nature of exercises that the student will visit (since the student can jump around using the Lesson Browser), each exercise node and its associated case nodes are added to the network dynamically at run-time when the student begins a new exercise.

Figure 4 shows an example portion of the network for the plot unit where the student has visited two exercises, as indicated by the "Exercise 1 Exploration" and "Exercise 2 Exploration" nodes. In the first exercise, the student was presented with a constant function, which has as relevant exploration cases positive intercepts (labeled "Exercise 1 Case 1" in Figure 4) and negative intercepts (labeled "Exercise 1 Case 2"). In the second exercise, the student was presented with a linear function, which has as relevant exploration cases positive intercepts, negative intercepts, positive slopes, negative slopes, and the zero slope (labeled "Exercise 2 Case 1" through "Exercise 2 Case 5" respectively). These cases are used to update the nodes representing the exploration of the related general concepts (labeled "PosIntercept Exploration", "NegIntercept Exploration", "PosSlope Exploration", "ZeroSlope Exploration" and "NegSlope Exploration") and the student's exploration of each individual exercise.

The other type of nodes found in the network represent the student's knowledge of function--related concepts. The extent of the student's explorations is used in part to judge how well the student seems to understand the material. The exercises in the Switchboard allow the students to demonstrate their knowledge directly, although these are the only exercises with any notion of 'correctness'.


I think you should explore a wider variety of graphs

You should see what the function looks like when it has a positive slope: pay close attention to the shape of the line!

You can modify the slope by typing a new value into the function equation text box. Think about the relationship between the function equation and graph!

Figure 7: The Hint Window

2.3 The Coach

In order to remain consistent with the philosophy of exploratory learning environments, it is crucial that the Coach supports student exploration as unobtrusively as possible. Thus, the Coach is designed to provide different levels of guidance, according to the needs of the individual learner.

The first level of guidance consists of a generic suggestion to continue exploring when a student tries to leave an exercise before having adequately explored it. Currently, the Coach does not interrupt a student's exploration of an exercise. Once the students signal that they wish to move to a different exercise, the Coach examines the students' behaviour in order to decide whether the exploration is satisfactory. In order to do so, the Coach queries the Student Model for two pieces of information: the probability that the student has adequately explored the current exercise and the probabilities for the relevant exploration concepts. The Coach remains silent if either the current exercise exploration is satisfactory (i.e. the related probability is above a pre-determined threshold), or if all of the associated exploration concept probabilities are satisfactory. If the student does not meet either of these two criteria, then a message is shown, suggesting that the student explore more and ask for hints if necessary. This message does not contain any concrete information as to what the student should explore, but includes a suggestion to ask for a hint. We omitted any specifics from this message to force the students to be as self-directed as possible in the exploration process, and to take initiative in obtaining hints. Since we want to maintain a high level of learner control, the student may always choose to disregard the Coach's suggestion and leave the exercise at any point. If a student does decide to stay, a suggestion is made to open the Exploration Assistant (currently only available for the Machine and Switchboard units), a tool that helps students monitor their exploration process by categorizing and displaying their recent exploratory actions. Figure 6 shows the tool open for an exercise in the Machine unit; it has organized the various inputs that the student has explored into relevant categories represented in the Student Model (such as Small-Positive-Range inputs, Zero inputs, etc).

As students explore, they can ask for a hint at any time. The Coach generates hints dynamically by traversing the concept hierarchies that are stored in the Student Model in a bottom-to-top, left-to-right manner. The traversal includes only those hierarchies that contain concepts relevant to the current exercise, and stops when an unexplored concept is found. Each applicable concept has a direct mapping to a hint object that contains a template for a progression of suggestions on how to explore that concept; these suggestions start off very general, and become more specific. Once an unexplored concept is found, its corresponding hint object is used to generate the hint. The Student Model continues to assess the student's actions between hint requests, and so the concept traversal is performed every time a hint is requested; this approach allows the Coach's suggestions to remain consistent with the current status of the student’s exploration.

We will illustrate the hint procedure by going back to the example in the previous section. Let’s suppose our student is working with a linear function, has already explored both positive and negative intercepts extensively, and has just requested a hint. In order to generate the hint, the Coach first traverses the exploration-related concept hierarchies that are relevant to the current exercise. In this exercise, the traversal begins with the hierarchy containing nodes related to intercept exploration (see “Intercept Exploration” nodes in Figure 4). Since the positive-intercept, negative-intercept and general intercept exploration node probabilities are satisfactory, the Coach moves on to examine the slope-concept hierarchy, starting with the “PosSlope Exploration”. This concept has a probability below the satisfactory threshold, and so the traversal stops here. The hint object linked to this node is used to generate the hint. Our student requests two more hints in succession; the hint window containing all three levels of hints related to slope exploration is shown in Figure 7. Currently, each concept has two to three levels of hints associated with it; further testing is required to determine the optimal number of levels.

3. Empirical Evaluation of ACE

3.1 Experimental design

The target population for ACE is high school students who are beginning to learn about functions. Thus, to evaluate if and how the current version of ACE influences students’ learning, we were planning to run a study with grade 11 students from a local school. Unfortunately, due to unforeseen last-minute scheduling difficulties with the school, we were unable to carry out the study with these subjects and had to resort to first year undergraduate students in our university. We only accepted subjects who were not currently taking any math courses, nor had done so within the past year. Nonetheless, several of our subjects showed very good function knowledge. Because only 14 subjects signed up for the study and because several subjects showed a ceiling effect in the pre-test, a two-groups design was unlikely to give any reliable information on ACE. Thus, we decided to use all of the 14 subjects in one experimental group, to gain an initial understanding of how and if system usage affects learning.

The one-session study was carried out in our computer science research lab. Each student used ACE for 30 minutes. To gauge students’ learning, we gave them an equivalent paper-and-pencil pre-test and post-test. The tests consisted of 39 questions equally divided into categories of function output recognition and generation, graph property recognition, equation property recognition, and equation–graph correspondence. In addition, the students wrote a 9-item questionnaire to assess their subjective experience with ACE.

Each session was observed by one of the experimental team members, who recorded data on standardized observer sheets. ACE itself also produced log files of the students’ interactions. From these files, we extracted a number of interaction events, including: 1) number of exercises passed (a student ‘passed’ an exercise if the Student Model indicated sufficient exploration); 2) total number of exploration hints requested; 3) average level of hint accessed by each subject; and 4) total exploratory actions performed. In the next section, we report results from the analysis of the log files, questionnaire and observer sheets.

3.2 Results

Effect of ACE on learning. We first wanted to verify if interaction with ACE triggered any learning at all. It did, as we found a statistically significant difference ($p = 0.013$) between the pre-test average (78.4%) and post-test average (92.3%), despite the fact that 8 out of our 14 subjects had very high pre-test scores. Second, we wanted to understand how system usage influences learning. Thus, for each of the event counts extracted from the log files, we ran a regression analysis with that event count and pre-test scores as independent variables, and post-test scores as the dependent variable[1]. Pre-test score was always a significant positive predictor of post-test scores.

We found the following positive predictors of post-test scores (after controlling for pre-test)

1. Total number of exploration hints accessed [$p = 0.0406$, R^2=84.6%].
2. The number of exercises passed [$p = 0.0093$, R^2= 87.9%].

These results provide an initial indication that ACE's support of the exploratory process does improve learning. The first result confirms that some students do need help when interacting with an open learning environment. The second result also suggests that the Student Model accurately predicts when students are ready to move to a new exercise, because ACE lets students leave an exercise without warning only when the Student Model assesses that they have adequately explored it. The above results could, of course, also be caused by additional factors (such as student's general academic ability or conscientiousness) that might influence the related event counts and post-test performance. The fact that there is no correlation between event counts in 1 and 2 above suggests that this is not the case, but only a formal study can show this more reliably.

The total number of exploratory actions that students performed was not a significant predictor of learning. This might be due to a tendency that we noticed in several students to "over-explore". When these students received a hint from the system to stay and explore more, they stayed, but then tended to try every available case, even those related to concepts that they had already explored. Redundant explorations likely did not contribute to improving the student's understanding, which explains the lack of correlation between number of exploratory actions and learning. Over-exploration is consistent with one of the problems students have in open learning environments: the inability to monitor one's own progress during the exploration process [15].

The system's Exploration Assistant is specifically designed to help students monitor their exploration, but not a single subject used it, possibly because it was relegated to the tool bar and labeled with an ambiguous icon. This requires either re-designing the interface so that the Exploration Assistant is more accessible, or having ACE explicitly suggest its use whenever over-exploration occurs.

Another relevant result obtained from the log files showed that the number of times a stay event was generated was a positive predictor of the number of exploratory actions performed [$p = 0.0378$, R^2= 31.2%]. At first, this may seem like an obvious result: a suggestion is made to explore an exercise further, it is followed, and thus more exploratory events are generated. However, it is possible that a subject may choose to stay, inspect the interface without performing any meaningful actions, and then move on to the next exercise – this in fact did happen a number of times with one subject. Nonetheless, the fact that stay events were typically followed by exploratory actions indicates that ACE's interventions are successful at encouraging exploration.

[1] Due to our small sample size, we were only able to include at the most two independent variables in our model.

Subjects' Perception of ACE. In general, students' answers to the questionnaire indicate that they enjoyed using the system and found it useful. The degree to which subjects found the hints helpful (measured on a scale from 2 to -2) was a positive predictor of their post test scores [$p = 0.0339$, $R^2 = 85.0\%$]**,** after controlling for the pre-test**.** This indicates that the subjects' questionnaire answers were not simply dictated by a desire to please (a common confounding variable in subjective questionnaires) but reliably reflected their opinion.

We also found that the average level of hints requested were predictors of the degree to which subjects found the hints helpful [$p= 0.0267$, $R^2 = 34.7\%$]. This indicates that although in many cases the first, generic level of hint was sufficient to trigger more exploration (74% of all hints requested were at the first level), the more detailed hints were useful to the people who needed them.

Further qualitative observations from the observer sheets. All subjects traversed the curriculum sequentially – in fact, only one tried "jumping around", and even then only towards the end of the session. The fact that a "Next exercise" button was considerably more accessible than the Lesson Browser doubtlessly encouraged this behaviour; making the Lesson Browser more visible or available would likely encourage a less linear approach to the curriculum.

Although we did find a positive relationship between learning and the number of hints used, hints were not requested as often as subjects seemed to need them. A number of subjects indicated that they had forgotten about the hints – this suggests that the interface should emphasize that hints are available. On the other hand, we also believe that some students simply have a tendency not to ask for help. These students either flounder, in which case the system should react to long pauses and 'wandering', or they move on without learning the required concepts. In such cases the system should intervene more aggressively, until it becomes apparent that the student has taken charge of their own learning; this will have to be investigated in further studies.

Finally, by observing the students' interaction with the system, we realized that even individuals who know the material and explore adequately need reassurance at times that they are in fact "doing the right thing". This type of support is not related to either domain- or exploration-specific knowledge provision, but rather is a form of emotional support for the students. Currently, the system indicates when the student has explored enough by saying "good job" when the student asks to move to the next exercise, but says nothing as the student is exploring. In future versions, we will explore ways of identifying students who require more verbose support, and the means of providing it.

4. Conclusions and Future Work

We have presented a prototype intelligent exploratory learning environment, ACE (Adaptive Coach for Exploration), whose goal is to provide tailored adaptive support to student exploration. ACE aims to address one of the main limitations of open learning environments: that students who do not already possess the capability to learn through autonomous and unconstrained exploration generally do not learn as effectively from these environments.

The approach we took to overcome this limitation involves three steps: providing students with highly graphical tools designed to encourage the exploration of domain concepts (related to mathematical functions in the current application); monitoring the student's exploration, to allow a probabilistic Student Model to assess the effectiveness of the student's exploratory process; using the Student Model's assessment to direct the interventions of an exploration

Coach. The Coach provides both unsolicited encouragement to explore more and hints on demand, which are tailored to improving the effectiveness of students' exploratory behaviour. Particular emphasis placed on guiding students who do not take the initiative to explore.

We described a preliminary study to evaluate ACE's effect on learning. The study shows that ACE does trigger learning, as seen in a significant increase in test scores following usage. Regression analyses of posttest scores on different interaction events and pretest score suggest that, as subjects explored the system more and asked for more hints, their learning increased. Subjects who requested hints in greater depth found them useful. The study also uncovered various ways to make ACE's interface and tailored support more effective, which we will investigate in future versions of the system.

Since the study we conducted did not have a control group, the results we report do not tell us how relevant ACE's tailored support is to triggering students' exploration and learning – perhaps the same results could be obtained with the ACE interface alone. To address this issue, we are planning to conduct a more formal study with grade-11 students who represent our initial target population.

Future work on ACE includes designing additional activities to encourage students' exploration. We will also focus on improving the ACE's student modeling in two ways. First, we would like to use eye-tracking technology to add data on user attention to the evidence used to assess exploratory behaviour. The second enhancement involves enriching the model's representation of exploratory behaviour by including additional user's features that influence this behavior, such as motivation and relevant meta-cognitive skills (e.g., self-explanation). This will enable the model to diagnose the causes of poor exploration and to support tutorial interventions that specifically target these causes. Finally, we plan on researching alternative ways of motivating exploration, which could supplement ACE's hints and suggestions.

References

1. Aleven, V., K.R. Koedinger, and K. Cross. *Tutoring answer-explanation fosters learning with understanding*. in *AIED '99, 9th World Conference of Artificial Intelligence and Education*. 1999. Le Mans, France.
2. Anderson, J.R., *et al.*, *Cognitive Tutors: Lessons Learned*. The Journal of the Learning Sciences, 1995. **4**(2): p. 167-207.
3. Collins, A. and J.S. Brown, *The computer as a tool for learning through reflection*, in *Learning issues for intelligent tutoring systems*, H. Mandle and A. Lesgold, Editors. 1990, Springer: New York.
4. de Jong, T. and W. van Joolingen, R., *Scientific Discovery Learning With Computer Simulations of Conceptual Domains*. Review of Educational research, 1998. **68**(2): p. 179-201.
5. Koedinger, K.R., *et al.*, *Intelligent tutoring goes to school in the big city*, in *Proceedings of the 7th World Conference on Artificial Intelligence and Education*, J. Greer, Editor. 1995, AACE: Charlottesville, NC. p. 421-428.
6. Lesgold, A., *et al.*, *Sherlock: A coached practice environment for an electronics troubleshooting job.*, in *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches*, J.H. Larkin and R.W. Chabay, Editors. 1992, Lawrence Erlbaum Associates: Hillsdale, NJ. p. 201-238.
7. Njoo, M. and T. de Jon, *Exploratory Learning with a Computer Simulation for Control Theory: Learning Processes and Instructional Support*. Journal of Research in Science Teaching, 1993. **30**(8): p. 821-844.

8. Paolucci, M., D. Suthers, and A. Weiner, *Automated advice-giving strategies for scientific inquiry*, in *Intelligent Tutoring Systems: Proceedings of the Third International Conference*, C. Frasson, G. Gauthier, and A. Lesgold, Editors. 1996, Springer: Berlin. p. 372-381.

9. Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 1988, San Mateo, CA: Morgan-Kaufmann.

10. Shute, V.J., *A comparison of learning environments: All that glitters...*, in *Computers as Cognitive Tools*, S.P. Lajoie and S.J. Derry, Editors. 1993, Lawrence Erlbaum Associates: Hillsdale, NJ. p. 47-73.

11. Shute, V.J. and R. Glaser, *A large-scale evaluation of an intelligent discovery world.* Interactive Learning Environments, 1990. **1**: p. 51-76.

12. Shute, V.J. and J. Psotka, *Intelligent tutoring systems: Past, Present and Future*, in *Handbook of Research on Educational Communications and Technology*, D. Jonassen, Editor. 1996, Scholastic Publications.

13. Stewart, J., *Caculus: Single Variable, Early Transcendentals*. 3rd ed. 1995, Pacific Grove: Brooks/Cole.

14. van Joolingen, W. and T. de Jong, *Supporting hypothesis generation by learners exploring an interactive computer simulation*. Instructional Science, 1991. **20**: p. 389-404.

15. van Joolingen, W.R., *Cognitive Tools for Discovery Learning*. Journal of Artificial Intelligence in Education, 1999. **10**.

16. VanLehn, K., *Conceptual and meta learning during coached problem solving*, in *ITS96: Proceeding of the Third International conference on Intelligent Tutoring Systems.*, C. Frasson, G. Gauthier, and A. Lesgold, Editors. 1996, Springer-Verlag: New York.

17. Veermans, K. and W.R. van Joolingen. *Using induction to generate feedback in simulation-based discovery learning environments*. in *ITS '98, 8th International Conference on Intelligent Tutoring Systems*. 1998.

18. White, B., T. Shimoda, and J. Frederiksen, *Enabling students to construct theories of collaborative inquiry and reflective learning: computer support for metacognitive development.* International Journal of AI in Education, 1999. **10**: p. 151-182.

19. White, B., *ThinkerTools: Causal models, conceptual change and science education*. Cognition and Instruction, 1993. **10**(1): p. 1-100.