The Adaptive Coach for Exploration: An Intelligent Open Learning Environment

Heather Neilson
heatherneilson@gmail.com

Department of Computer Science
University of British Columbia
2366 Main Mall
Vancouver, B.C. Canada V6T 1Z4

1. Introduction

Open learning environments are computer-based systems that promote student exploration of a domain rather than explicit instruction (Bunt, Conati, Huggett & Muldner, 2001). They allow students to explore new concepts at their own pace and without the constraints of a question-and-answer format. With this approach, the student takes on a bigger role in her own learning (de Jong & van Joolingen, 1998); it is hoped that this will promote deeper understanding of the material (Bunt, Conati, Huggett & Muldner). Not all students benefit from an open learning environment though. Students must take the initiative to thoroughly explore the concepts; in open learning environments, students often explore only a relatively small portion of the available possibilities (Kuhn, 1992). Students must also have certain meta-cognitive skills in order to learn by exploration (Njoo & de Jong, 1993; Shute & Glaser, 1990); some students have difficulty generating hypotheses (de Jong & van Joolingen). As such, the skill of self-explanation is valuable for students using an open learning environment; a student self-explains by reasoning about the solution to an example problem (Conati & VanLehn, 2000). The Adaptive Coach for Exploration (ACE) aims to capture the benefits of open learning while providing tailored support to guide students in their exploration and encourage them to self-explain (Bunt & Conati, 2002; Bunt, Conati, Huggett & Muldner).

2. Description of ACE

ACE is an interactive open learning environment for exploration of mathematical functions; it employs a Student Model and a Coach to provide intelligent support. The Student Model dynamically represents the learning behaviour of a student and the knowledge that she has. The Coach uses the Student Model to generate appropriate hints and guide the student through the exercises at the right pace.

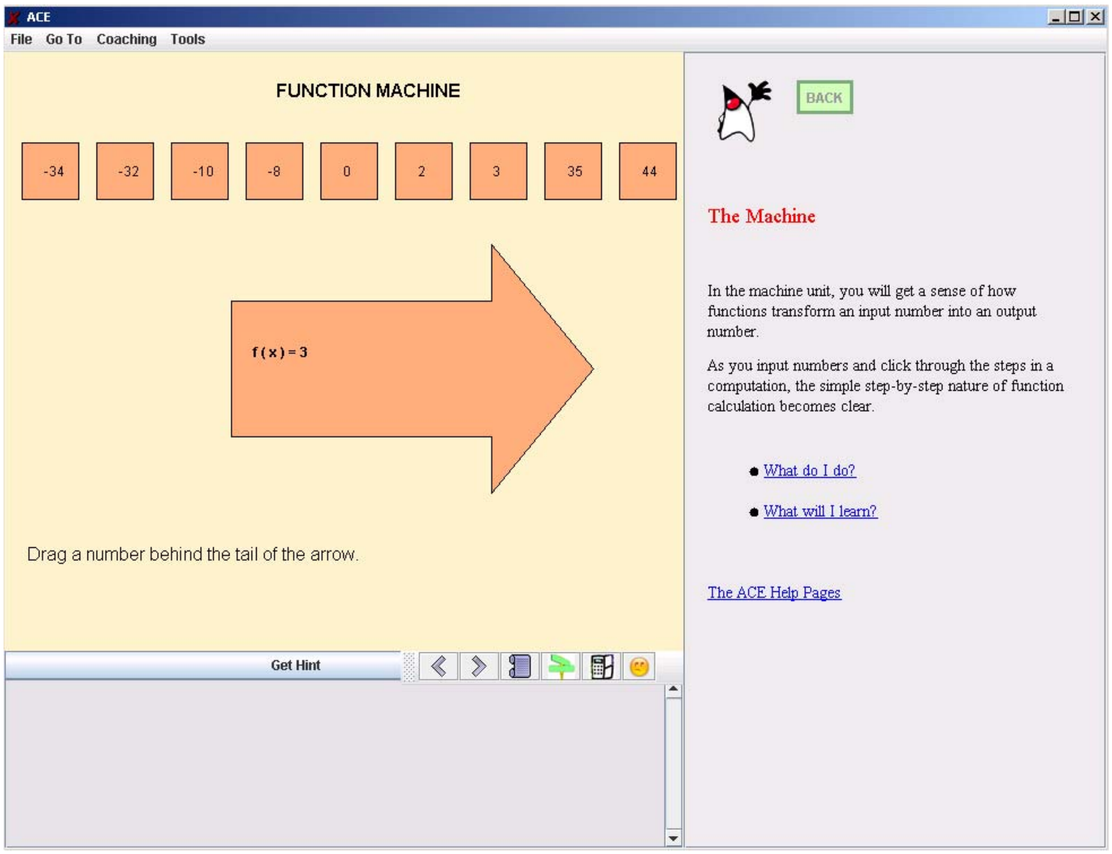
2.1 The Graphical User Interface (GUI)

The ACE GUI has three main window areas: the graphics window, the feedback window and the help window. The interactive exercises are displayed in the graphics window. Hints and tips are displayed in the feedback window, and a button is available to get a hint at any time. Next to the hint button is a toolbar for moving between exercises and opening a variety of tools, which are discussed at the end of this section. The help window explains how to use the program, what to do in each unit and what the learning goals are in each unit.

There are three units in ACE: the Machine Unit, the Arrow Unit and the Plot Unit. In the Machine Unit (Figure 1), the student drags and drops different inputs into the function machine. The machine substitutes in the input, and the student clicks to see each step in the solution. In the Arrow Unit (Figure 2), a list of possible inputs and a list of possible outputs are displayed along with a function equation. With the help of a calculator tool, the student connects inputs to their corresponding outputs. In the Plot Unit (Figure 3), an equation and its graph are displayed. The student explores the function by dragging the graph around and observing the changes in the equation, or by entering new values into the equation and observing the changes in the graph.

Each unit contains a number of exercises, defined in a curriculum file. Exercises can be about constant, linear, power or polynomial functions. The student can move through the exercises in sequential order, or she can jump to any exercise using the Lesson Browser tool in the toolbar.

In addition to the ‘Next Exercise’ and ‘Previous Exercise’ buttons and the Lesson Browser tool, the toolbar contains buttons for opening the Calculator, the Exploration History, and the Self-Explanation dialog box. The student can use the Calculator for the computations needed in the Arrow Unit, and the Exploration History shows a list of inputs or concepts she has explored (see Figure 4). The Self-Explanation dialog box allows the student to answer a multiple-choice question about a concept of her choice. This tool encourages the student to test her self-explanations.



Drag a number behind the tail of the arrow.

The Machine

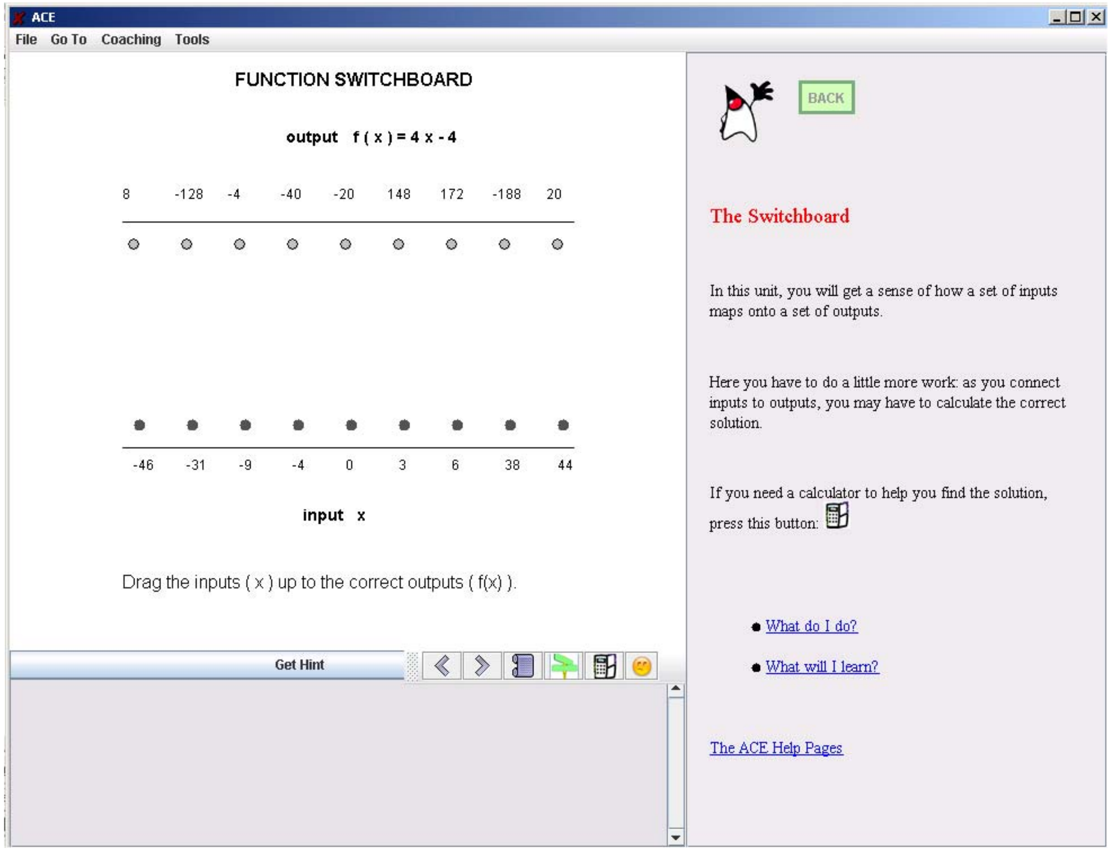
In the machine unit, you will get a sense of how functions transform an input number into an output number.

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

- What do I do?
- What will I learn?

The ACE Help Pages

Figure 1: In the Machine Unit, the student plugs different inputs into the machine to see the function solution.



Drag the inputs (x) up to the correct outputs (f(x)).

The Switchboard

In this unit, you will get a sense of how a set of inputs maps onto a set of outputs.

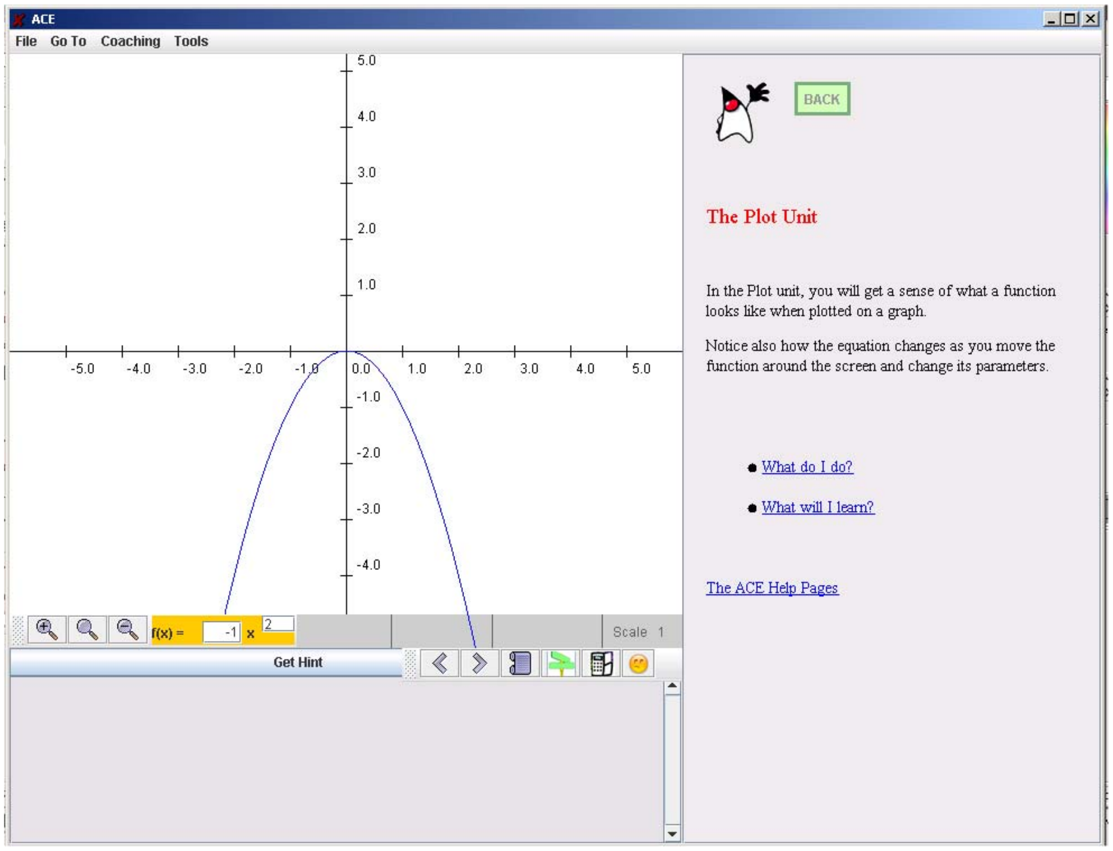
Here you have to do a little more work: as you connect inputs to outputs, you may have to calculate the correct solution.

If you need a calculator to help you find the solution, press this button:

- What do I do?
- What will I learn?

The ACE Help Pages

Figure 2: In the Arrow Unit, the student connects inputs to outputs for the given function.



The Plot Unit

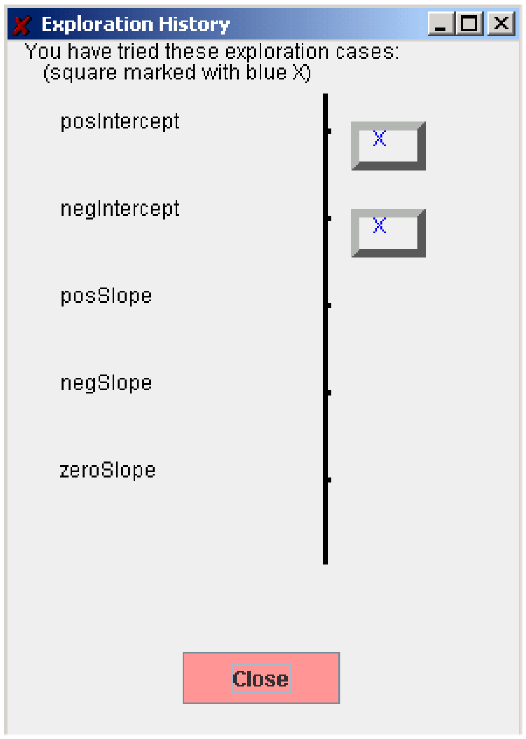
In the Plot unit, you will get a sense of what a function looks like when plotted on a graph.

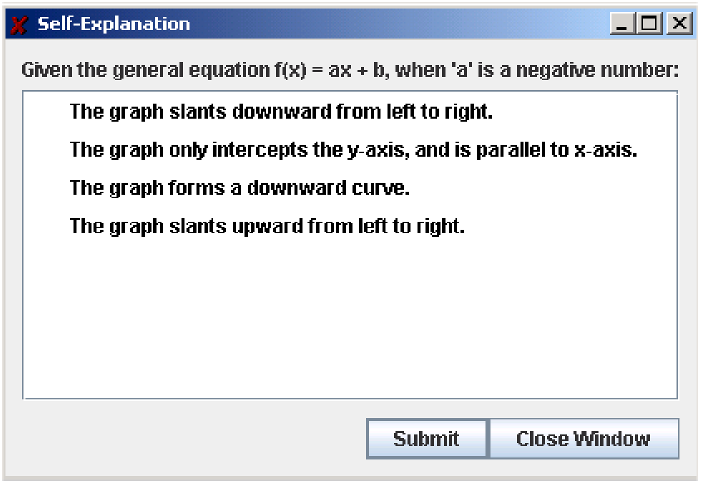
Notice also how the equation changes as you move the function around the screen and change its parameters.

- What do I do?
- What will I learn?

The ACE Help Pages

Figure 3: In the Plot Unit, the student explores the relationship between a function and its graph.

a. 

b. 

Given the general equation f(x) = ax + b, when 'a' is a negative number:

The graph slants downward from left to right.

The graph only intercepts the y-axis, and is parallel to x-axis.

The graph forms a downward curve.

The graph slants upward from left to right.

Figure 4: The exploration history (a) indicates which concepts have been explored so far. The Self-Explanation dialog (b) allows the student to test her hypotheses.

2.2 The Student Model

The Student Model uses a Bayesian network to represent the student's knowledge and behaviour (for an introduction to Bayesian networks, see Heckerman & Wellman, 1995). The Model is used to assess the student's learning patterns and generate appropriate hints. When the student performs an exploratory action, such as putting an input into the function machine or moving a function graph, the Student Model is updated.

The network contains several different types of nodes. The knowledge nodes reflect correct actions in the Arrow Unit; if the student correctly matches an input with an output, it is taken to be more likely that the student understands the relevant concept. The exploratory nodes represent the student's exploration at different levels of granularity. Exploration is assessed at the levels of individual exercises, concepts, function types and units. Concept exploration is hierarchically organized; the positive slope exploration, negative slope exploration and zero slope exploration nodes are all linked to the slope exploration node. Each exercise has an associated set of relevant exploration cases, depending on the current unit. In the Machine Unit and the Arrow Unit, the relevant exploration cases correspond to the different types of available inputs (small positive inputs, large positive inputs, zero, small negative inputs and large negative inputs). In the Plot Unit, the relevant exploration cases correspond to exploration of the different properties of the function. For example, exploration cases for a linear function include negative slope and positive intercept, and exploration cases for a power function include odd exponent and positive shifting. The Student Model's assessment of exercise exploration and concept exploration is based on evidence from the relevant exploration case nodes. That evidence also affects a node representing the student's tendency to self-explain. See Figure 5 for a diagram showing some of the network connections.

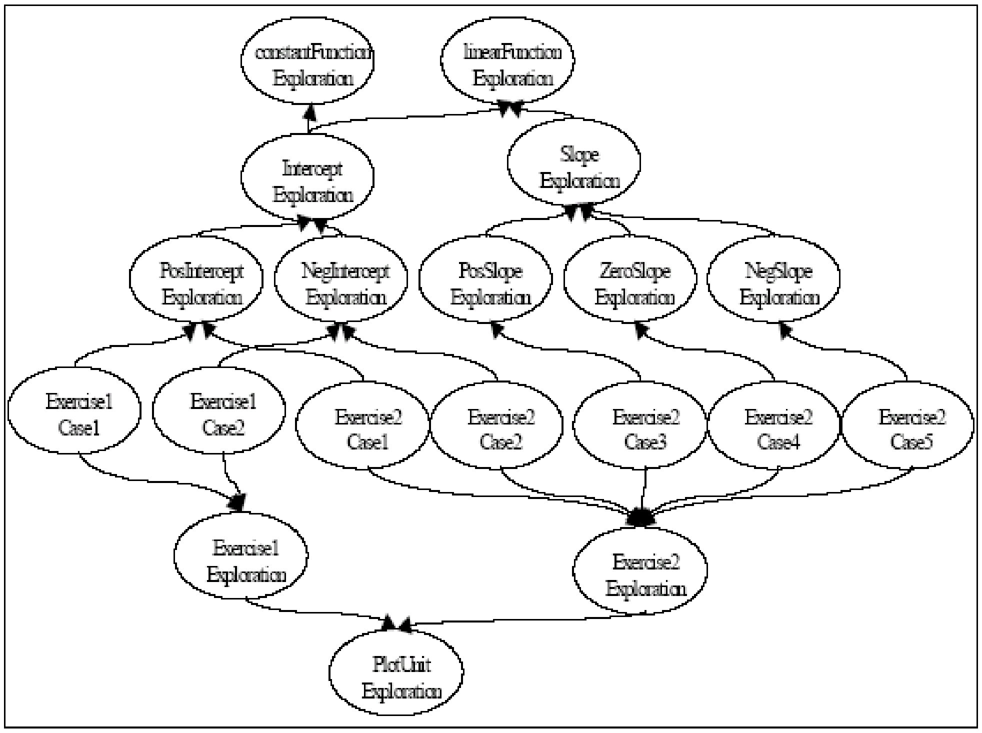


Figure 5: Node organization in the Student Model. (Figure taken from Bunt, Conati, Huggett & Muldner, 2001.)

2.3 The Coach

The Coach queries the Student Model to generate appropriate hints and guidance. If the student requests a hint, the hint given will be determined based on which concepts have not been thoroughly explored. When the student chooses to move on to another exercise, the Coach will check whether the exercise has been properly explored (the exercise exploration node is compared with a threshold value). If the exercise has not been explored well enough, the Coach will suggest that she should stay and get a hint.

In the Plot Unit, the Coach also provides occasional prompts to encourage the student to self-explain. After each exploratory action, a decision is made as to whether a prompt should be given. This decision is based on the student's tendency to self-explain (if the Model indicates that the student is very likely to self-explain on her own, a prompt will not be generated) and the student's exploration so far of the concept currently being explored. This information is represented by the tendency to self-explain node and the appropriate concept node, respectively.

Two types of prompt can be given – gentle prompts that are less intrusive, and strict prompts that are more intrusive. A gentle prompt (Figure 6) is a highlighted message that appears on the graph or above the equation, suggesting that the student should think carefully about the relationship between the equation and the graph. The first prompt given for any concept is a gentle prompt; if another prompt is needed for that concept after a later exploratory action, a strict prompt is given. When a strict prompt is given, the student is asked to consider how the graph would look given a particular change in the equation (related to the concept she is currently exploring). The student chooses one of several possibilities and then she is given an indication of whether she chose correctly (see Figure 7). The prompts are designed not to interfere with the student as long as she is generating explanations on her own, and to encourage self-explanation if she is not.

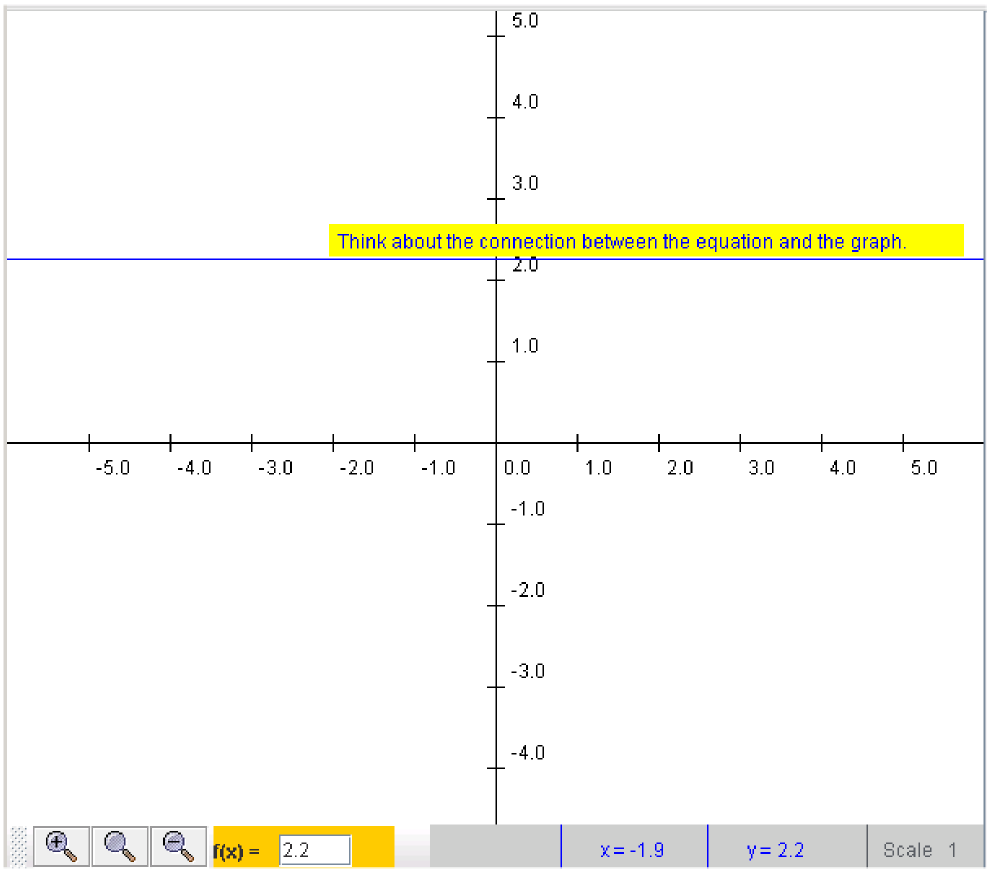
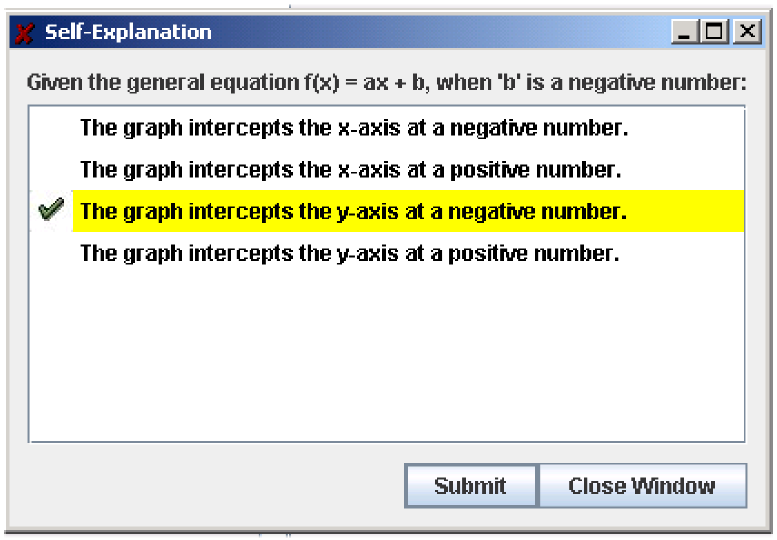


Figure 6: An example of a gentle prompt that is given to encourage the student to self-explain.



Given the general equation f(x) = ax + b, when 'b' is a negative number:

The graph intercepts the x-axis at a negative number.

The graph intercepts the x-axis at a positive number.

✓ The graph intercepts the y-axis at a negative number.

The graph intercepts the y-axis at a positive number.

Figure 7: An example of a strict prompt that is given to guide the student to explain the concept of negative intercepts.

3. My Contributions to ACE

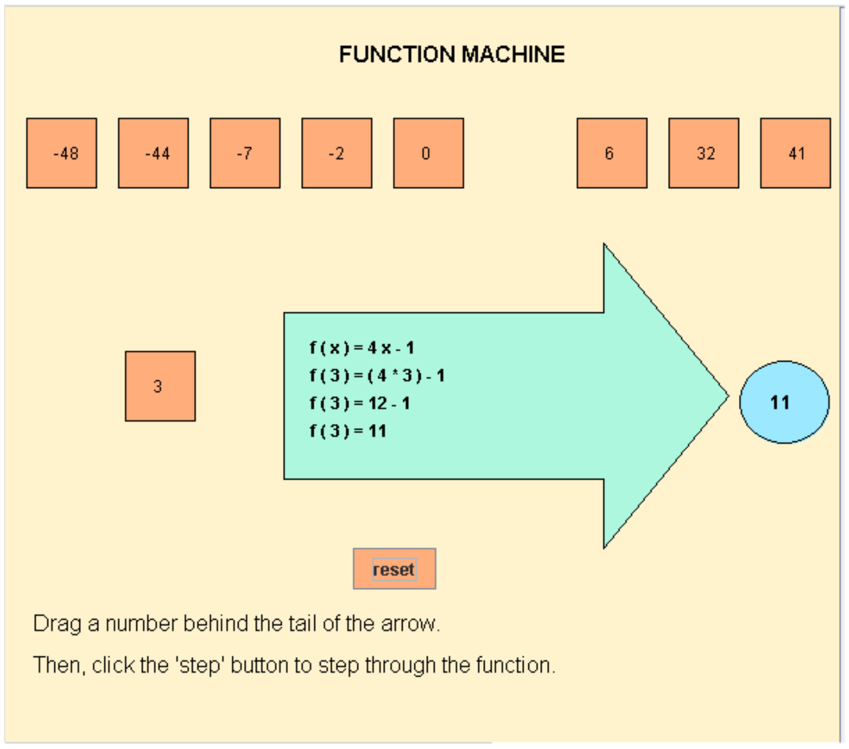
My contributions to ACE were mostly concerned with improving the interface by keeping it in line with the goal of providing the level of support that the student needs while minimizing intrusiveness (Bunt, Conati, Huggett & Muldner, 2001) and making it more user-friendly. I also created new algorithms to determine when to give the student a prompt to self-explain, and to select an appropriate hint when the student asks for one. This section will highlight some of the most important changes made and describe the new algorithms for generating prompts and hints.

3.1 Interface Improvements

When the student requests a hint, she should receive a clear and helpful message. The set of hint messages was updated to include increased variation in wording and sentence structure in order to minimize repetition. It is hoped that this will help to maintain the student's interest and avoid annoyance with the hints. For the same reason, instead of giving exactly the same message each time a gentle prompt pops up in the Plot Unit, a message is randomly selected from a set of different messages each time a gentle prompt is given.

A balance is important between minimizing intrusiveness and providing enough supportive, directive and reinforcing feedback. Several changes were made to reduce ACE's intrusiveness. For example, if the student is prompted to answer a question but she closes the dialog box without answering, the same question will not be asked again after her next action, even if she still has not thoroughly explored the concept. Previously, when the student moved on to a new exercise (and the Coach determined that she was ready to do so) a "Great Work!" message would pop up. This message provided positive feedback, but it also stopped the flow of the program, requiring the student to click 'OK' to continue. In this case, priority was given to reducing the number of popups, and the message was removed for the current version.

The previous version of the Machine Unit interface displayed only one step in the equation solution at a time. After each click to see the next step, the previous step disappeared and the new step reappeared in its place. In order to allow the student to pay more attention to solution process rather than remembering the previous step in the solution, the interface was updated to instead display each step underneath the last one (see Figure 8).



f(x)=4x-1
f(3)=(4*3)-1
f(3)=12-1
f(3)=11

Drag a number behind the tail of the arrow.

Then, click the 'step' button to step through the function.

Figure 8: Each step in the function solution is displayed in the function machine.

3.2 The Prompt Algorithm

Each time the student performs an exploratory action an exploration case node is created. These nodes are stored in the case history, as a record of the student's exploration; they also contribute evidence to the Student Model, by influencing the appropriate concept node. After every exploratory action, the algorithm depicted in Figure 9 is carried out to determine whether the student should be prompted to self-explain.

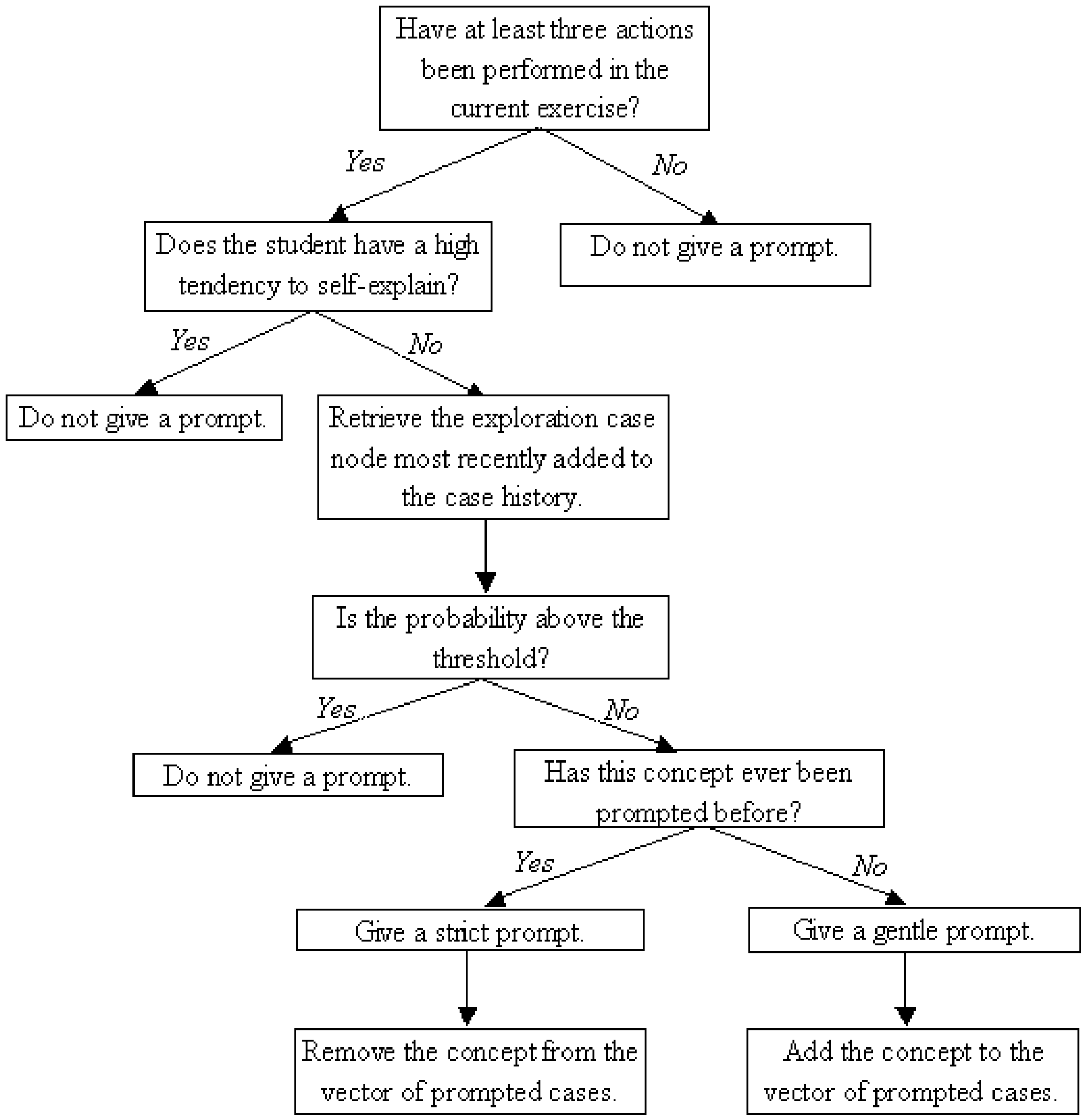

Have at least three actions been performed in the current exercise?
Yes
No
Does the student have a high tendency to self-explain?
Do not give a prompt.
Yes
No
Do not give a prompt.
Retrieve the exploration case node most recently added to the case history.
Is the probability above the threshold?
Yes
No
Do not give a prompt.
Has this concept ever been prompted before?
Yes
No
Give a strict prompt.
Give a gentle prompt.
Remove the concept from the vector of prompted cases.
Add the concept to the vector of prompted cases.

Figure 9: A schematic representation of the algorithm for determining whether a prompt should be given.

3.3 The Hint Algorithm

The hint algorithm is called when the student requests a hint. It begins by checking whether the student is currently stepping through a function in the Machine Unit. If she is, a hint is given telling her to click ‘step’ or enter a new input. Otherwise, if the current exercise exploration node indicates that the exercise has already been thoroughly explored, a hint is given that suggests moving to the next exercise.

In addition to the above-mentioned special cases, there are three levels of hints; level zero is the most general, and level two is the most specific. Each time the user starts a new exercise the level is set to zero. A level-zero (general) hint is given at most once per exercise; subsequent hints alternate between level one and level two such that one medium and one specific hint are given for each unknown concept.

The topic of the hint is determined by a function that queries the Student Model to find out which of the concepts relevant to the current exercise and unit have not been thoroughly explored yet. These (unexplored) concepts are stored in a vector, such that the hints given cycle through in a rotation.

4. Future Developments

ACE is an intelligent open learning environment that provides tailored support for students as they explore mathematical functions. It is geared towards helping students learn the domain-general skill of self-explanation; as such, the Student Model and Coach components of ACE could be incorporated into learning tools for other subject areas with only superficial changes.

References

Bunt, A. & Conati, C. (2002). Assessing effective exploration in open learning environments using Bayesian Networks. *Proceedings of ITS 2002, 6th International Conference on Intelligent Tutoring Systems*. Biarritz, France, 698-707.

Bunt, A., Conati, C., Huggett, M. & Muldner, K. (2001). On improving the effectiveness of open learning environments through tailored support for exploration. *Proceedings of AIED 2001, 10th World Conference of Artificial Intelligence and Education*, San Antonio, TX, U.S.A., 365-376.

Conati, C. & VanLehn, K. (2000). Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. *International Journal of Artificial Intelligence in Education*, 11, 398-415.

de Jong, T. & van Joolingen, W.R. (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research*, 68, 179-202.

Heckerman, D. & Wellman, M.P. (1995). Bayesian Networks. *Communications of the ACM*, 38, 27-30.

Kuhn, D., Schauble, L. & Garcia-Mila, M. (1992). Cross-domain development of scientific reasoning. *Cognition and Instruction*, 9, 285-327.

Njoo, M. & de Jong, T. (1993). Exploratory learning with a computer simulation for control theory: Learning processes and instructional support. *Journal of Research in Science Teaching*, 30, 821-844.

Shute, V.J. & Glaser, R. (1990). A large-scale evaluation of an intelligent discovery world: Smithtown. *Interactive Learning Environments*, 1, 55-77.