In *Proc. of Tenth International Conference on AI in Education,* May 2001.

An Automated Lab Instructor for Simulated Science Experiments

Aaron D'Souza,[1] Jeff Rickel,[1] Bruno Herreros,[2] and W. Lewis Johnson[1]
[1] *USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA, 90292*
adsouza@usc.edu, rickel@isi.edu, johnson@isi.edu
[2] *USC Department of Chemistry, Los Angeles, CA, 90089-1062*
herreros@chem1.usc.edu

Abstract. Virtual laboratories, which allow students to interactively run simulated experiments, are a powerful way to teach students about science. To provide guidance to students working with virtual labs, we developed an automated lab instructor (ALI). ALI has a representation of the key relationships in the simulation model that the student should learn, and it uses this knowledge to interleave its teaching opportunistically with the student's own discovery learning. Specifically, it can recognize learning opportunities in a student's experiments, test the student's resulting understanding, and gently guide the student towards these learning opportunities when necessary. ALI has a simple yet general ontology for representing knowledge of simulation models, as well as domain-independent tutorial capabilities derived from this ontology, so it can be connected to a new virtual lab with relatively little effort.

1 Introduction

Virtual laboratories are a powerful way to teach students about science. A typical virtual lab provides simulation models of physical phenomena (e.g., the kinetic molecular theory of gases), animations that allow students to visualize simulations (e.g., bouncing balls in a container representing gas molecules), and a user interface that allows students to run simulated experiments (e.g., vary temperature to study the resulting effect on the gas). As with a physical laboratory, students learn about scientific methods by playing the role of a scientist, and they learn the principles governing phenomena through their own experience, rather than by being told. Moreover, virtual labs nicely complement physical labs. Students can study phenomena that occur on too large or small a spatial or time scale to permit physical experiments. Their ability to visualize phenomena that would otherwise be unobservable (e.g., individual gas molecules) helps them form useful mental models. Finally, since they only need a computer, rather than physical lab equipment, they can run experiments practically anywhere and anytime.

Virtual labs are becoming widely available. New science textbooks increasingly come with such software, and many virtual labs are available for free on the Internet. However, while these labs allow students to interactively run experiments, few provide any guidance. Without guidance, students may run experiments of little instructional value, and they may fail to draw appropriate conclusions from successful experiments. However, requiring the presence of a human lab instructor would eliminate the ability of students to run experiments

anywhere and anytime, while dictating an exact sequence of experiments for students to run and questions for them to answer would prevent them from using their own interests and questions to drive their experiments.

To address these issues, we developed an automated lab instructor (ALI) that provides flexible guidance to students interacting with virtual labs. ALI has a representation of the key relationships in the simulation model that the student should learn, and it is capable of explaining these relationships as well as testing the student's understanding. The central principle behind our approach is that ALI should teach these relationships in the context of the student's own experiments; ALI interleaves its teaching opportunistically with the student's own discovery learning. To support such a tutorial style, ALI must recognize learning opportunities in a student's experiments, test the student's resulting understanding, and gently guide the student towards these learning opportunities when necessary.

While designing such a tutor for a particular virtual lab is challenging in itself, our goal is more general: it should be easy to connect ALI to a new virtual lab and to give ALI the knowledge it needs to interact with students using that lab. To achieve this goal, ALI has a simple yet general ontology for representing knowledge of simulation models, as well as domain-independent tutorial capabilities derived from this ontology. In addition to providing such knowledge to ALI, a course author need only define a few simple interface functions to connect ALI to an existing virtual lab.

2 Related Work

Our work integrates tutorial dialogue for scientific inquiry with interactive simulation. The seminal research in the first area, tutorial dialogue for scientific inquiry, was performed by Stevens, Collins, and Goldin [1, 16]. They introduced several key representations for knowledge of physical phenomena; our knowledge representation is based on Qualitative Process Theory [2], which is a direct intellectual descendant of their work. However, like other similar work (e.g., [5]), their approach is based solely on a natural language dialogue between the tutor and student; it does not incorporate the use of interactive simulations that can provide an experimental testbed for students, which is a central element of our approach.

In contrast, most tutoring systems that have exploited interactive simulations have supported little or no dialogue with students to guide their scientific inquiry. STEAMER [7] introduced the notion of interactive, inspectable simulations to help students build mental models of physical phenomena. We share that goal but seek to combine graphical simulations with a tutorial dialogue in which the student and computer tutor can investigate simulation models together. More recently, Forbus and Falkenhainer developed methods for "self-explanatory simulators" [3]. Such simulators combine quantitative simulation with knowledge represented using Qualitative Process Theory to allow the computer to provide a running commentary on what is happening during the simulation, and to answer questions about the simulation afterwards. This is a powerful foundation for intelligent tutoring, but, unlike ALI, self-explanatory simulators have no explicit pedagogical objectives to drive the dialogue. Recent work has explored an ambitious combination of such simulators with tutoring components for scientific inquiry [8], but only as a proof of concept; no general integration has been done. Other interactive simulators have been integrated with a tutorial dialogue (e.g., [11, 13, 18]), but those dialogues focus on teaching procedures for operating and maintaining physical systems, not scientific inquiry for understanding them.

Several other recent systems are exploring tutorial goals similar to ours, but they appear to lack the generality and reusability we are targeting because they are hard-coded for particular domains rather than exploiting more general knowledge representations as we propose. ISIS [9] provides a simulated ecosystem in which students can run experiments, as well as a tutor that offers advice on formulating research questions, generating hypotheses, designing and running experiments, and drawing appropriate conclusions. LUCID [17] provides interactive simulations with which chemistry students can explore experimental questions and solve problems. Their focus is complementary to ours; while we focus on tutorial dialogue with a computer tutor, they focus on supporting teams of students working together to provide feedback to one another.

3 Learning Environment

At the heart of our learning environment is a simulation of some physical phenomena. A simulation is characterized by a set of independent variables and a set of dependent variables. Independent variables are those that the student is free to modify, while dependent variables are ones whose values are computed from the independent variables by the simulation. The goal of our learning environment is to have the student learn the relationships between the independent and dependent variables by running experiments in the virtual laboratory. With each experiment, the student initializes the independent variables, runs the simulation, and observes the effect on the dependent variables.

The learning environment runs as a Java applet in a web browser, as shown in figure 1. The upper section of the GUI allows the student to control the simulation, including starting and stopping the simulation, changing the independent variables, and viewing the values of dependent variables.

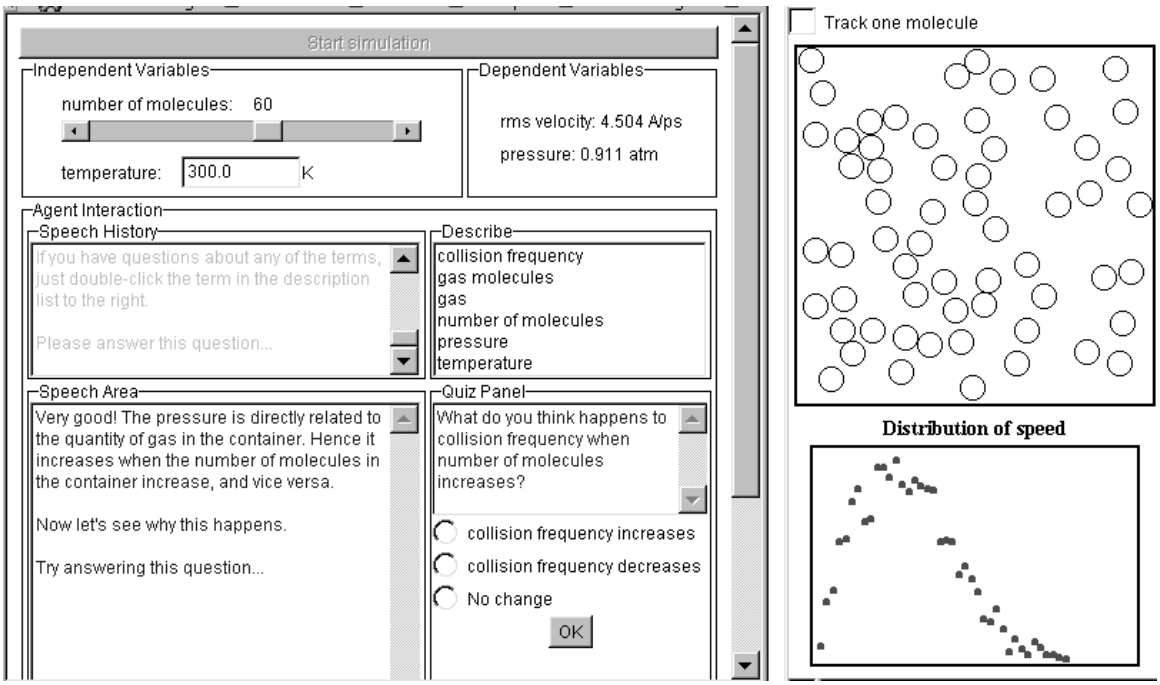

Figure 1: The learning environment as seen by the student.

The frame to the right contains an animation that is optionally supplied by the simulation author. This provides any appropriate visual cues about the operation of the simulation. In the example in figure 1, the animation shows the motions of gas molecules within a container, and a histogram of the molecules' speeds.

The *Agent Interaction* section of the GUI supports ALI's dialogue with the student. The *Speech Area* displays ALI's comments. As new text appears in the speech area, old text is moved into the *Speech History* area so that the student always has access to ALI's prior comments for reference. To the right of the speech area is a *Quiz Panel* in which ALI occasionally asks multiple-choice questions to test the student's understanding.

Because ALI may introduce terms that the student is not familiar with, the student should be able to ask ALI to explain them at any point during the interaction. To the right of the speech history area is a list of terms that the student can select and ask ALI to describe. This list is updated whenever ALI introduces a new term.

4 Creating a New Virtual Laboratory

Figure 2 shows the architecture of our learning environment. There are three components: ALI, ALI's domain-independent interface manager, and a domain-specific simulator. A course author can connect ALI to a new simulator in two simple steps: (1) write a set of methods that provide the interface between ALI's interface manager and the simulator, as described in section 4.1; and (2) describe the simulation domain using ALI's knowledge specification language, as described in section 4.2.

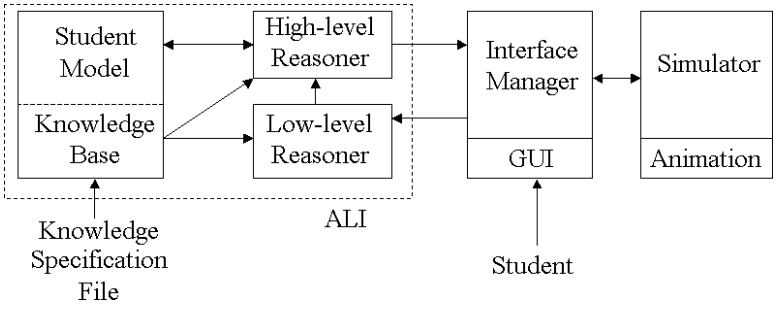

Figure 2: General architecture of the learning environment.

4.1 Interface manager API

The interface manager creates and manages the student's GUI, and it connects the simulator to ALI. It provides a simple interface to the simulator, including methods that the interface manager invokes, which must be implemented by the simulator, and methods provided by the interface manager, which are invoked by the simulator. The simulator must implement three methods: `getSimulationVariableInfo`, `startSimulation`, and `stopSimulation`. The `getSimulationVariableInfo` method, which is called when ALI is initialized, should return descriptions of each of the independent and dependent variables. The interface manager uses this information to maintain an internal state of the simulation and to automatically generate the GUI through which the student runs experiments. The interface manager informs the simulator when the student clicks on "Start simulation" or "Stop

simulation" by calling the `startSimulation` or `stopSimulation` methods, respectively. The `startSimulation` method can request the current values of the independent variables from the interface manager.

Given the values of the independent variables, the simulator computes the resulting behavior of the system being simulated. It can either model a dynamic process in which the system reaches equilibrium over a series of timesteps, or perform a static computation of the final system state. As the simulation updates its variables, the new values are communicated to ALI using an event-generation interface provided by the interface manager. At each timestep in the simulation, it generates a `SimulationVariableEvent` to notify the interface manager of updates. A single event can encapsulate the updates to all the variables in this timestep, and thus suffices to convey the updated state of the entire simulation.

4.2 Knowledge specification language

To provide guidance to the student, ALI must know the learning objectives of the simulation. Our goal in designing ALI was to help students acquire a basic qualitative understanding of physical phenomena. Students often develop the ability to solve the quantitative equations governing physical phenomena without acquiring such a basic understanding. Virtual labs can complement lectures and textbook exercises by helping students to acquire better mental models of the phenomena behind the equations. Thus, our goal was to design a knowledge specification language with which a course author can easily represent the qualitative understanding of a simulation model that students are expected to acquire through the virtual lab.

Our language is based on the influences among state variables in a simulation model. Such a representation has been used by human modelers in a wide variety of domains [4, 12, 15], and it has also received considerable attention in research on automated reasoning about physical systems [2, 14]. The details of our language are drawn from Qualitative Process Theory (QPT) [2], which provides a qualitative abstraction of differential equations. The current implementation supports QPT's functional influences among state variables, Q+ and Q-, defined as follows:

$$x \overset{Q+}{\rightarrow} y \quad \Leftrightarrow \quad y = f(x, \dots) \text{ and } \frac{\partial y}{\partial x} > 0$$

$$x \overset{Q-}{\rightarrow} y \quad \Leftrightarrow \quad y = f(x, \dots) \text{ and } \frac{\partial y}{\partial x} < 0$$

This form of modeling ignores the exact functional relationship between the two variables and simply specifies the direction of change that should result in one variable as a result of a change in the other (when all other variables are held constant). ALI does not yet support QPT's differential influences, I+ and I-, which model the influence of one variable on the rate of change of another variable. Thus, our current knowledge specification language can model the steady state (equilibrium) relationships between independent and dependent variables, but not the transient (dynamic) relationships.

Course authors provide ALI's knowledge in a text file that uses structured definitions for each of the items in the knowledge base. To illustrate each type of item, we will use examples from an implemented simulation that teaches the kinetic molecular theory of gases.

```
Entity {name: gas; description: The gas molecules in the container;}
```

An entity in the simulation is any object that can possess properties. Entities are used as placeholders that tie the other knowledge base objects such as properties and influences into a cohesive whole. The names of entities are also placed in the "Describe" panel of the GUI so that students can get descriptions of them.

```
Property {name: number of molecules; ofEntity: gas;
   description: The number of gas molecules in the container;}
Property {name: collision frequency; ofEntity: gas;
   description: The rate at which the molecules collide with the
                container walls;}
Property {name: pressure; ofEntity: gas;
   description: The force exerted by the gas on the container walls;}
```

Variables in the simulation model are represented as properties of entities. Properties do not necessarily have to correspond to explicit variables in the simulation. For example, the second property in the above specification corresponds to a variable that is not explicitly computed in the simulation engine, but is simply used to illustrate (as described below) a lower-level influence between the number of molecules in the system and the collision frequency of the molecules with the container walls, which explains the high-level relation between the number of molecules and observed pressure.

```
Relation {name: nMolecules_pressure; type: Q+;
   properties:  number of molecules, pressure;
   description: The pressure is directly related to the quantity of gas
                in the container. Hence it increases when the number of
                molecules in the container increase, and vice versa;
   explanation: nMolecules_freq, freq_pressure;}
```

Relations represent the high-level influences in the simulation. In this example, a positive influence exists between the independent variable `number of molecules` and the dependent variable `pressure`. The `properties` field lists the pair of variables involved in the influence. The `type` field specifies the type of influence, currently Q+ or Q-. The definition also implicitly assumes causality from the first property to the second, as in QPT.

To understand a physical phenomenon, students must learn not only how it behaves (represented by the high-level influences) but also why it behaves that way. Typically, a high-level influence between two variables can be explained by a more-detailed chain of influences connecting those variables. The `explanation` field specifies a list of lower-level influences that provide such an explanation chain. These low-level influences are called `AtomicRelations`.

```
AtomicRelation {name: nMolecules_freq; type: Q+;
    properties:  number of molecules, collision frequency;
    description: The number of collisions of gas molecules with the
                 container is proportional to the number of gas
                 molecules present. Hence an increase or decrease in
                 the number of molecules results in a corresponding
                 increase or decrease in the frequency of collisions;}
AtomicRelation {name: freq_pressure; type: Q+;
    properties:  collision frequency, pressure;
    description: Collisions of the gas molecules with the container
                 walls result in the pressure exerted by the
                 gas. Hence an increase in the frequency of these
                 collisions increases the pressure and vice versa;}
```

`AtomicRelation` structures are very similar to the `Relation` structures discussed above, but lack the `explanation` fields. These knowledge elements form the building blocks of the explanation lists of `Relation` objects. In this example, the two `AtomicRelation` objects form the explanation list for the `nMolecules_pressure` influence.

5 ALI

5.1 Overview

The interface manager tracks changes in the simulation at every timestep, but ALI reasons about the state of the simulation at only two points: the start and end of a student's experiment. The decision to have the reasoning mechanism triggered at these two points was made for efficiency reasons; having ALI reason about the operation at every timestep was wasteful, and could produce erroneous decisions if the system being simulated had not yet converged to a stable state. Having the simulation send updates at every timestep, however, is necessary for the interface manager to display updated variable values in the GUI. Also, other data analysis utilities could be integrated into the interface manager, such as graphing and statistical analysis, and these would require a complete record of variable values. An additional benefit of tracking state changes at every timestep is that we could use such information to calculate the rate of change of variables, which would be necessary to model QPT's I+ and I- influences.

ALI's reasoning about influences spans across two consecutive runs of the simulation. This allows a change in independent variables and its influence on the system to be determined by comparing it against the previous time the simulation was run. A student that changes a single independent variable at a time can isolate the effect of that variable on the dependent variables. If a student changes multiple independent variables at once, ALI will use the opportunity to provide instruction in the scientific method, and caution the student against such modifications.

To simplify the discussion, the next two subsections will assume that ALI's model of the student is in its initial state (i.e., the student has not yet begun experimenting with the simulation). A discussion of the effect of the student model on ALI's interaction is deferred to section 5.4.

5.2 Low-level reasoning

Each influence in the knowledge base represents a learning objective. The task of the low-level reasoner is to identify the instances when an influence is illustrated during the student's experiments. These instances are learning opportunities that ALI exploits to engage the student in a dialogue about the influence. When a simulation is stopped, the low-level reasoner scans the list of influences and, for each influence, determines whether its independent and dependent variables have changed in a manner consistent with its influence type (Q+ or Q-) since the last time the simulation was stopped. If so, the low-level reasoner sends the high-level reasoner a "knowledge event" notifying it of the specific influence that was illustrated.

The low-level reasoner also scans for anomalous changes in dependent variables, such as a pressure value that increased in response to a decrease in temperature, when in fact it should have decreased according to the influence in the knowledge specification. Such anomalous

readings may confuse the student, and are hence reported to the high-level reasoner so that ALI may comment on them.

5.3 High-level reasoning

The high-level reasoner determines ALI's behavior as a function of the student model (see section 5.4) and the knowledge events generated by the low-level reasoner. This phase of reasoning uses the knowledge base to construct an interaction based on the current event. Let us take a simple example to illustrate the manner in which such interactions are created.

Assume that as the student experiments with our gas simulation, ALI receives an event indicating that the gas pressure increased as a result of the student increasing the number of molecules in the system. By virtue of its existence in the knowledge base, this influence is one of the learning objectives of the simulation.

To ensure that the student noticed this learning opportunity, ALI will generate a quiz. The quiz asks the student if he/she noticed the effect on pressure that resulted from the change in the number of molecules. The quiz does not have to be authored; ALI has quiz templates into which the names of appropriate properties are added from the knowledge base to construct the complete quiz. Quizzes are multiple-choice questions of the following form:

> Did you notice what happened to pressure when the number of molecules was increased? (pressure increased, pressure decreased, did not notice)

If the student answers incorrectly or did not notice the change, ALI will suggest that the student try again:

> Why don't you try increasing the number of molecules again, and this time keep a close eye on pressure.

Otherwise, since this is the first time that the student has encountered this influence, ALI will describe the reason for this observed effect on pressure. This description is taken from the knowledge base element corresponding to the influence:

> Very good! The pressure is directly related to the quantity of gas in the container. Hence it increases when the number of molecules in the container increases. Now let's see why this happens . . .

After describing the influence, ALI will engage the student in a dialogue about its underlying causes. It traverses each link in the explanation chain for this influence, and asks the student a question about each link in turn. For our example influence, the two low-level relation links in the explanation chain are the influence of the number of molecules on their collision frequency with the container walls, and the influence of this collision frequency on the perceived pressure of the gas. Thus, for the first link, ALI would pose the following question:

> What do you think happened to the collision frequency when the number of molecules was increased? (collision frequency increased, collision frequency decreased, no change)

At the same time, ALI knows that the term "collision frequency" is new and places it in the list of describable elements in case the student needs to have it elaborated. Next, ALI gives the student feedback on his/her answer ("yes" or "no") followed by a textual description of the influence. For example, if the student answered the above quiz incorrectly, ALI would respond as follows:

No. The number of collisions of gas molecules with the container is proportional to the number of gas molecules present. Hence an increase or decrease in the number of molecules results in a corresponding increase or decrease in the frequency of collisions.

Regardless of the student's answer, ALI will proceed through the remaining explanation elements:

What do you think happens to pressure when the collision frequency increases? (pressure increases, pressure decreases, no change)

After discussing each link in the explanation chain, ALI summarizes all the low-level influences in the context of the high-level influence being explained:

In summary . . .

- An increase in the number of molecules increases collision frequency
- An increase in collision frequency increases pressure

Hence an increase in the number of molecules increases pressure. Why don't you try running some more experiments . . .

Besides identifying opportunities for instruction, ALI also keeps track of how productive the student's actions are in terms of uncovering situations for learning new relations. If the student has run several experiments without encountering any new learning opportunities, ALI will suggest an experiment based on the learning objectives the student hasn't mastered. For example, after the student has mastered the relationship between the number of molecules and pressure, ALI will expect him to explore the relationship between temperature and pressure. If the student doesn't perform any experiments that demonstrate this influence, then after a few experiments ALI will suggest one:

Why don't you try increasing temperature?

5.4 Student Model

ALI keeps a record of the student's presumed knowledge, for two reasons. First, ALI must be able to recognize when the student has accomplished all the learning objectives for a simulation, as well as guide the student towards unachieved learning objectives. Second, ALI must distinguish between new information presented to the student and references to material that has already been discussed; without making this distinction clear to students, they may tune out repeated material and miss new information [10].

The student model is stored as an overlay model [6] that encapsulates the information within the state of the knowledge base objects. The attributes are represented as simple flags: whether or not the knowledge element has been encountered, whether it has been explained, and whether the question relating to an influence was answered correctly or incorrectly. These attributes could be serialized out to permanent storage as a model of each student, as well as a record of past sessions which can be used to determine ALI's level and nature of interaction with the student during future sessions or with subsequent models.

As an example of the effect of the student model on ALI's interaction, we return to our gas law simulation. When our student encounters the influence between the number of molecules and pressure again, ALI will check to see if this relation has been understood. If it hasn't, then ALI will proceed to go through the explanation chain again. This time, however, ALI will only describe the reason for an influence if the student gets an answer wrong. If ALI described an influence previously, and the student's answer demonstrates a correct understanding of the influence, there is no reason to repeat the description.

When walking a student through the explanation chain of a high-level influence, ALI may encounter a low-level influence that was already described in the explanation of a different high-level influence. In such a situation, ALI modifies its text to clearly identify the new discussion as referring back to a prior discussion:

Do you remember what happens to pressure when the collision frequency is increased?

At the end of each explanation, ALI will check to see if all the learning objectives for the simulation have been covered. If so, it will summarize the main learning objectives, and let the student know that he/she has completed all of them. Otherwise, it will suggest that the student continue running further experiments.

6 Evaluation

We conducted a small, formative evaluation to assess ALI's usability and effectiveness. Six students from a freshman chemistry course at USC filled out a short survey on their background, took a short pre-test on the kinetic molecular theory of gases, interacted with ALI for 15-30 minutes, and then took a post-test identical to the pre-test and filled out a questionnaire on ALI. They were given no instructions on using ALI except that ALI would tell them when they were done. After filling out the questionnaire, we discussed their opinions with them. Their pre- and post-tests were graded by the third author of this paper, who holds a Ph.D. in chemistry.

Although no strong conclusions can be drawn from such a small study, it suggests that ALI is easy to use and does improve students' understanding. All the students were able to use ALI successfully without any human guidance, and they all rated ALI as easy to use. (According to the questionnaire, all of them use computers frequently for a variety of tasks, so we cannot draw conclusions about ALI's usability for less computer-literate students.) The answers on the pre-tests indicated that students already had a basic understanding of the kinetic molecular theory of gases (they had begun covering it in class earlier in the week), so there were few substantial improvements on the post-test. Nonetheless, the answers on the pre-tests tended to be more intuitive and mechanistic, while the answers on the post-test focused more on the relations between independent and dependent variables and were generally more accurate. Ideally, we would prefer that the relational perspective augment, rather than replace, the mechanistic view, which suggests that ALI should engage the students in more dialogue to relate the variables and relations to the visual aspects of the animation. This is an important area for future study, since it lies at the heart of combining tutorial dialogue with graphical simulations.

To test the generality of ALI, we evaluated additional science simulations developed by California State University, to assess the ease of applying ALI to them. We chose one simulation in particular for a trial integration: EvolutionLab, which simulates the evolution of

finches on desert islands in response to changes in environmental conditions. We found that a variety of equilibrium relationships in EvolutionLab could be modeled with ALI's ontology. In addition, other relationships could be modeled if we extended ALI to include QPT's differential influences (discussed in section 4.2) and its "views," which would allow ALI to recognize qualitatively important state changes. While equilibrium models form a large, important class of simulations, incorporating more of QPT into ALI to further increase its generality is an important area for future work.

7 Conclusions

ALI provides guidance to students interacting with virtual labs. ALI's interaction with a student is driven by specific pedagogical goals, and its guidance is interleaved with the student's own discovery learning. These pedagogical goals are based on the qualitative influences within a physical system that the student should develop an intuition for. Our design makes it easy to connect ALI to a new virtual lab and provide the knowledge that ALI needs to interact with students. Thus, ALI makes it possible for students to learn about a wide range of science topics through their own experiments, practically anywhere and anytime.

References

[1] Allan Collins and Albert L. Stevens. Goals and strategies of inquiry teachers. In R. Glaser, editor, *Advances in Instructional Psychology*, volume 2. Erlbaum Associates, Hillsdale, NJ, 1982.

[2] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[3] Kenneth D. Forbus. Using qualitative physics to create articulate educational software. *IEEE Expert*, 12(3), 1997.

[4] Jay W. Forrester. *Principles of Systems*. Wright-Allen Press, Cambridge, MA, 1968.

[5] Reva K. Freedman. *Interaction of Discourse Planning, Instructional Planning and Dialogue Management in an Interactive Tutoring System*. PhD thesis, Northwestern University, 1996.

[6] Ira P. Goldstein. Overlays: A theory of modelling for computer-aided instruction. Artificial Intelligence Laboratory Memo 495, Massachusetts Institute of Technology, Cambridge, MA, 1977.

[7] James D. Hollan, Edwin L. Hutchins, and Louis Weitzman. Steamer: An interactive inspectable simulation-based training system. *AI Magazine*, 5(2):15–27, 1984.

[8] Kenneth R. Koedinger, Daniel D. Suthers, and Kenneth D. Forbus. Component-based construction of a science learning space. *International Journal of Artificial Intelligence in Education*, 10:292–313, 1999.

[9] Thomas N. Meyer, Todd M. Miller, Kurt Steuck, and Monika Kretschmer. A multi-year large-scale field study of a learner controlled intelligent tutoring system. In *Proceedings of the Ninth International Conference on Artificial Intelligence in Education*, pages 191–198. IOS Press, 1999.

[10] Johanna D. Moore. Making computer tutors more like humans. *Journal of Artificial Intelligence in Education*, 7(2):181–214, 1996.

[11] A. Munro, M.C. Johnson, Q.A. Pizzini, D.S. Surmon, and D.M. Towne. Authoring simulation-centered tutors with RIDES. *International Journal of Artificial Intelligence in Education*, 8:284–316, 1997.

[12] C.J. Puccia and R. Levins. *Qualitative Modeling of Complex Systems*. Harvard University Press, Cambridge, MA, 1985.

[13] Jeff Rickel and W. Lewis Johnson. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13:343–382, 1999.

[14] Jeff Rickel and Bruce Porter. Automated modeling of complex systems to answer prediction questions. *Artificial Intelligence*, 93:201–260, 1997.

[15] Nancy Roberts, David Andersen, Ralph Deal, Michael Garet, and William Shaffer. *Introduction to Computer Simulation*. Addison-Wesley, Reading, MA, 1983.

[16] A. Stevens, A. Collins, and S. Goldin. Misconceptions in students' understanding. In D. Sleeman and J.S. Brown, editors, *Intelligent Tutoring Systems*, pages 13–24. Academic Press, 1982.

[17] Troy Wolfskill and David Hanson. LUCID: Computer-based activities for in-class team learning. www.chem.sunysb.edu/hanson-foc/lucid.html.

[18] Beverly Woolf, Darrell Blegan, Johan H. Jansen, and Arie Verloop. Teaching a complex industrial process. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 722–728, Los Altos, CA, 1986. Morgan Kaufmann.