Developing A Hidden Domain for Human and Electronic Students

Jeremy Ludwig, Alex Davis
Stottler Henke Associates
San Mateo, CA
{ludwig, davis} @stottlerhenke.com

Mark Abrams, Jon Curtis
Cycorp
Austin, TX
{mabrams, jonc} @cyc.com

Abstract

One objective of DARPA's Bootstrapped Learning (BL) program is to develop a general electronic student (e-student) that makes use of machine learning algorithms to learn from the kind of focused instruction typically provided by a human teacher. Over the course of the program, such a student was developed and trained against a number of learning challenges codified in electronic curricula spanning disparate domains of interest. In order to test the capabilities of this e-student, a "hidden domain" curriculum was created. An electronic version of this curriculum was used to test the e-student by an independent research group. Additionally, university undergraduate and graduate students were trained and tested using a human-accessible version of this same "hidden domain" curriculum, allowing researchers to gauge the performance of humans on the curriculum and use that performance as a benchmark against which to evaluate the e-student. This paper provides details on the hidden domain curriculum, which teaches a procedure for the diagnosis and repair of a satellite ground control station. We also present as a summary of the available human and e-student results.

1 Introduction

The Bootstrapped Learning (BL) program is a DARPA initiative aimed at advancing the state of the art in instructable computing. One objective of this program is to create an *electronic student* (e-student) that can learn from the kind of focused instruction typically provided by a human teacher to a human student [Oblinger, 2006]. The e-student uses a variety of machine learning algorithms to learn how to solve tasks in an arbitrary domain, where the teacher instructs with a set of formally defined natural instruction methods (NIM). For example, in using the *teaching by example* NIM the instructor may make gestures at relevant problem attributes, demonstrate actions in the domain, and provide explanations for why actions were taken. More concretely, an unmanned aerial vehicle operator might instruct the e-student on how to recognize transferring packages between trucks by showing the system specific examples of the activity, as well as noteworthy counterexamples, and annotating these examples to point out the most important aspects [Oblinger 2006]. This type of machine learning has significant implications for the development of intelligent applications in nearly any domain [Beal at al., 2009]. Instead of learning from large amounts of data or having computer programmers work with subject matter experts to manually encode knowledge in an expert system, an e-student would learn from a limited amount of instruction provided directly by the expert.

There are a number of research areas being investigated under the multi-year, multi-organization BL program. Roughly divided, one team is charged with developing the e-student [e.g., Natarajan et al., 2010] while another develops the curricula that the e-student is expected to learn, an electronic teacher that "executes" the curricula, and a framework[1] that supports interactions between the two electronic agents. The two teams work together to define the formal language and the natural instruction methods used by both teams (either in teaching the curricula or learning from them). This paper focuses on the curriculum development aspect of the process, describing the "hidden domain" curriculum that played a significant role in the BL program.

Curricula fill two roles in the BL program – to instruct the e-student and to test the e-student to see if it learned the concepts correctly. While it is envisioned that a human would teach the e-student directly, a defined curriculum has several advantages in algorithm development, e-student testing, and developing a repository of curricula that can be utilized by other researchers.

Each curriculum is organized in the same manner, called a ladder, where simpler concepts are taught in early rungs and more complex concepts that build upon the simpler ones are taught "higher up the ladder," in later rungs. Each rung in the ladder teaches a single concept, though this concept may be supported by multiple lessons, each (typically) conforming to a single NIM. For example, a single concept (such as how to recognize that a truck is parked) could be taught in three lessons: one by telling, one by example, and one that allows the student to practice what it has learned so far, and provides corrective feedback when necessary. Les-

[1] Henceforth referred to as the BL Framework.

sons can be taken by the student multiple times; in the case of 'by Example' and 'by Feedback' lessons, different "runs" through a lesson will result in, respectively, more and different types of examples, and different 'initial states' from which the student can begin practicing. Additionally, each curriculum contains background knowledge that is provided to the student. The background knowledge includes the basic concepts and procedures (typically, the basic commands for controlling objects in a simulated environment) from the domain that are required for learning to occur. That is, the basics are essentially programmed into the e-student, giving it "bare bones" upon which to build as it works its way up the ladder.[2] The curricula are formally defined, and instruction is automated. This allows for the algorithms behind the e-student to be developed and tested in an efficient manner, and for algorithm writers to experiment by varying formal parameters of the codified lessons.

Six curricula, spanning five distinct domains, were codified and made available to e-student developers over the course of the BL program. The first two, Blocks World and RoboCup, are domains commonly used as testbeds in artificial intelligence research. Three additional diversity domains have been constructed as part of the BL program to provide additional learning challenges: operating an unmanned aerial vehicle, planning (company-level) movement of armored vehicles, and diagnosis of errors on the International Space Station. [Ludwig et al., 2010; Natarajan et al., 2010], for which two curricula were developed.

This paper focuses on the development of a seventh curriculum, known as the "hidden domain." (HD) While the six so-called "diversity domain" curricula were used to train the e-student, the automated HD curriculum was used to test the e-student by an independent research group. Additionally, university undergraduate and graduate students were trained and tested in order to gauge the performance of human students on the same curriculum. In this paper we provide details on the hidden domain curriculum, which is the diagnosis and repair of a satellite ground control station, in both Phase II and III of the BL program. We also present as a summary of the human and e-student results.

The remaining subsections in the introduction cover the intended role of the hidden domain and background information on satellite ground control stations. After this, the Phase II and Phase III versions of the HD curriculum are described in detail. The paper ends with an overview of the human and e- student results and a brief conclusion. While, Phase II and Phase III have little meaning outside of the BL program they are used in this paper to describe the curriculum at two different points in time (roughly 2009 & 2010).

[2] This is not to say that the learning done in this program is purely compositional; novel concepts can be taught in a curriculum, so long as they apply to something about the simulated environment that serves as the shared domain of discourse between the eStudent and teacher.

1.1 Role of the Hidden Domain in the BL Program

The primary role of the hidden domain was to serve as a *phase gate* in Phase II of the Bootstrap Learning program. That is, for the research program to continue into the next year (Phase III) two particular conditions needed to be met. First, human students needed to pass the curriculum [Grant et al., 2011]. The human evaluation of the hidden domain curriculum was intended to demonstrate that the curriculum was necessary and sufficient to solve the final exam problems, providing evidence for the efficacy of the e-student curriculum. The human evaluation was also used to provide a baseline for expected e-student performance in the hidden domain. Human students were first presented with the domain background knowledge in a human-friendly format, then given a pre-test. Students who scored 20% or less on the pre-test were deemed "non-experts" and allowed to go through human student versions of all of the same lessons provided to the e-student. In order to pass, these students were expected to score 80% or better on the post-test (after training). The second condition is that the e-student pass the same two tests. The e-student was required to achieve a score of 0% on the pre-test based only on the pre-defined background knowledge and a minimum score on the final exam of 75% of the mean final exam score of the human students. For example, if human students averaged 91% then the e-student would need to score 68% to pass this condition.

The hidden domain curriculum was expanded in Phase III to include new, more difficult learning challenges. It was again tested on both human students [Grant et al., 2011] and the e-student. As was the case in Phase II, pre-test scores, based only on background knowledge, were required to be 20% or below for human students, and 0% for the e-student. No specific minimum achievement was required for the human students, but it was expected that the e-student receive a score on the final exam of 90% of the mean human score. Completion of the Phase III tests was one of the last milestones of the Bootstraped Learning program.

1.2 Hidden Domain Requirements

Selecting a hidden domain was actually quite difficult based on the constraints that were placed on hidden domain selection by the Bootstrap Learning program:

- The domain open to learning from experience. One of the primary ideas behind BL is that field users should be able to "teach" a system how to better do its job. Any system used in the curriculum should be expected to get smarter in the field.
- The system should actually NEED to learn from experience to succeed. That is, there should be some set of factors (e.g. environment) outside of the control of the system designers so that unexpected problems do occur.
- The domain needs to include explicit reasoning, as this is more amenable to BL than standard pattern recognition. In the UAV example above, the e-student is expected to learn the two-truck concept based on one or two examples, not thousands.

- The problem elements need to be easily visible to and encoded by an automated system.
- The domain should be one where problems that require diagnosis come up reasonably often in real life.

Additional constraints were imposed by the requirement for testing on both the e-student and human students:

- The domain needs to include problems that will challenge the human students. Average college students should not be able to solve the presented problems prior to training.
- But not too challenging - average college students should be able to learn how to solve the problems with only a few hours of learning.
- The exact same problems will be presented to human students and the e-student. Not only do the problems need to be challenging, yet learnable, by human students they also need to be challenging, yet learnable by the e-student.

The satellite ground station domain was seen to have the potential to meet all of these requirements.

1.3 Satellite Ground Stations

Satellites are monitored and controlled from mission control centers on the ground. The mission control centers communicate with the satellites through ground stations that operate satellite antennae. As satellites orbit the earth they periodically come within view of specific ground stations. The length of time that a satellite is potentially visible to a particular ground station depends on the altitude and inclination of the satellite's orbit and the location of the ground station. Satellites in low earth orbit circle the earth approximately every ninety minutes, and are in view of any particular ground station for short intervals of time, providing "windows" of ten to twenty minutes for potential contact. The hidden domain focuses on this type of satellite.

Ground stations operators (GSOs) operate the equipment that makes direct contact with the satellites. This includes the antenna, which must track the satellite's path as it moves across the sky, and the transceivers and other electronic equipment needed to encode and decode the information communicated. The equipment must be configured for each contact with each satellite. Normally, a set of configuration scripts is used to perform this configuration automatically. GSOs monitor the execution of the scripts and the transmissions between the ground station and the satellite, detect and diagnose any problems interfering with the ability to communicate, and repair the problems. Since contacts can be quite short, any problems must be detected and repaired quickly to salvage the contact. The precise software used to configure and monitor the connection varies among ground stations, depending on the organization operating the ground station and the nature of the satellites being contacted.

The hidden domain curriculum places the human or e-student in the role of a GSO. The student must monitor the configuration (during the pre-pass phase) and the operation (during the pass) of the ground station equipment and detect, diagnose, and repair any problems that occur. The curriculum makes use of a partial simulation of a particular software suite currently being used to communicate with some scientific satellites: the SatTrack suite developed by Bester Tracking Systems, Inc.

The curriculum focuses on problems in the downlink chain by which telemetry from the satellite is communicated to the mission control center via the ground station. Problems in the downlink chain are easier to diagnose than problems in the uplink chain (by which commands are sent from the mission control center to the satellite) because a simple heuristic applies that focuses the search for the source of the problem: the component closest to the source of the satellite's signal that does not exhibit normal status is the likely culprit. The scenarios found in the curriculum are based on real-world problems that have occurred to operators using SatTrack.

2 Phase II HD Curriculum

The Phase II curriculum consists of four units, broken down into a total of fourteen rungs. The first two units teach the learner to recognize anomalies, distinguishing between true anomalies and false positives, and to diagnose them. The rungs in the third unit each teach a different repair option. The final unit is a top-level procedure to diagnose and repair the ground station simulation.

The curriculum was designed to meet two specific constraints. First, unlike ground station operators who begin training with a college-level engineering education, the curriculum was designed to teach basic case-specific rules that could be learned in a few hours by a university student who possesses a lay understanding of diagnosis and devices. Second, the curriculum was designed to present specific learning challenges of interest to the e-student researchers.

In the remainder of this section we describe the background knowledge and simulator that support this curriculum, followed by a more in-depth examination of the curriculum and how the students were graded on it.

2.1 Background Knowledge

The background knowledge defines the basic concepts and relationships of the ground station domain. The information is presented as PowerPoint slides to the human students, and is manually encoded for the e-student.

The background knowledge defines the following concepts:

- **Components** – system, antenna, receivers, combiners, baseband processor, and decommutator
- **Indicators** – The measurements and/or settings available on each component and their possible values; for instance, the baseband processor has a *TLM Symbol Rate* indicator that can be set to a particular value
- **Component functions** – automated routines that can be performed on each component, such as *Reset* or *Track.*
- **Connections** – ways in which the components can be connected to one another to form a valid downlink path (Antenna → RF Receiver → Combiner → Baseband Processor → Decommutator); when a re-

dundant component fails it can be replaced with another

- **Passplan** – a list of the expected configuration settings for communication with the given satellite
- **Tasks** – a list of tasks that are completed, in order, to connect to and download information from the satellite; these tasks form a timeline of the pass and set the context for any errors that arise
- **Expectations** – a brief table that describes the expected state of the components based on the current tasks; some indicators are expected to show red (error) values at certain points in the task timeline that do not indicate any actual error

For the human student, this background knowledge is presented in the context of the ground station user interface (UI) that is used to view the *percepts* from the simulator (components, indicators, connections, passplan, tasks, and expectations) and to issue *commands* in the UI to the simulator (change connections, run component functions, set component values). In the case of the e-student, this background knowledge is manually encoded to allow the e-student to interact directly with the simulator to receive the same *percepts* and send *commands*.

2.2 Simulator

The ground station simulator provides a simulation of selected components of a satellite control ground station in communication with various satellites, and provides interfaces for students (both human and electronic) to view simulator percepts and issue commands. The UI for human use is modeled after the SatTrack application for ground station monitor and control, which is part of the SatTrack Suite of aerospace applications, commercially available and in use at the UC Berkeley ground station. A screenshot of the human UI is shown in Figure 1.

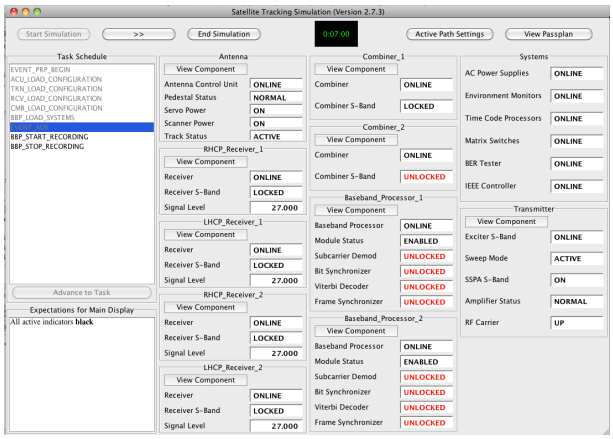

Figure 1. Ground station simulation main display for human students. Tasks are displayed in the upper left, expectations in the lower left, and status summary in the center. The toolbar is used to start and advance the simulation (top left) and view connections and the passplan (top right).

The simulator is implemented as a Java program and integrated with the BL Framework through the construction of Java interfaces and objects. Simulated states are translated into "percepts" – representations codified in the Interaction Language, the *lingua franca* for electronic agents that communicate via the BL Framework [Mailler et al., 2009]. The simulator updates component configurations and connections as percepts in real time over scenario execution, and accepts commands from the student including settings, connections, and function invocations.

The instructor is responsible for starting the simulator with a particular set of faults. There are six individual faults available in the simulator: antenna motion, intrack pointing, azimuth pointing, elevation pointing, baseband misconfiguration and baseband hangup. The faults can be combined to create multi-fault scenarios. While limited in number, each instance of a fault is quantitatively distinct. That is, all specific numbers associated with a fault are randomly generated within a pre-defined range, ensuring that the student will not see the exact same fault twice.

Once started, the passage of time as a context for diagnosis or recovery is represented by the state of the task list. Tasks are completed in order, and each task is marked as executed when it is finished. The human student progresses through the tasks by pressing a button to advance to the next task, while in the electronic version of the curriculum, the automated instructor advances the task for the e-student. Each diagnosis in the current curriculum may be performed in the context of a particular task having executed. Once the error is diagnosed, the student issues a series of commands to correct the error. To avoid telegraphing the correct solution to the student, the simulator includes many potential actions and component configurations that are not actually required for the available faults. These extra capabilities act as distracters to both the human and electronic students.

2.3 Curriculum Overview

This section briefly describes each of the units and rungs in the curriculum. Units are used to divide the laddered curriculum into smaller sets of related rungs for discussion purposes. The lessons are taught in a bottom up fashion where low-level concepts are taught initially, followed by higher-level concepts that build on the lower-level concepts.

For each rung, multiple lessons exist that adhere to the formally defined NIMs: by telling, by example, and by feedback. These lessons can include instructional messages—*utterances* (what the teacher "says") and *actions* (what the teacher does)—in addition to those generated as percepts by the simulator. For example, the instructor can *gesture* at a particular component of the satellite ground station and *state* that a property of that component is relevant to diagnosing the problem at hand.

A single research team was responsible for developing both a self-paced curriculum in PowerPoint for human students, and the automated curriculum for the e-student. The two versions of the curriculum were designed to be as close to one another as possible.

Unit 1: Identify Abnormal Components

Fault diagnosis revolves around abnormal components, which are components not operating as expected for a par-

ticular satellite pass. Each fault results in at least one abnormal component.

Unit 2: Fault Identification
Six rungs instruct the learner on how to identify the particular fault responsible for an abnormal component. Each rung identified a particular type of error that can occur: a component might be misconfigured, uninitialized, inappropriately offline; additionally, the antenna might be subject to a pointing error: an intrack error, elevation error, or azimuth error.

Units 3 and 4: Repair Procedures
The learner is taught how to fix each of the six possible faults in six rungs. If misconfigured, the problem can be resolved by setting the misconfigured control to its correct value. If initialized, the problem can be resolved by using the *reset* component function. If inappropriately offline, the problem can be resolved by changing the connections to make use of a redundant component. If the problem is a pointing fault, the problem can be resolved by changing one of the bias controls.

Note that this was broken into two distinct units for the e-student, where Unit 3 teaches only the correction procedures (e.g., how to correct a misconfiguration) and Unit 4 connects the fault to the correction procedure (e.g., if misconfigured then apply the procedure for correcting a misconfiguration). For the human students, these two units were combined to streamline instruction.

Unit 5: Top Level Procedure
The final unit in the curriculum contains a single rung that teaches a top-level, diagnose-and-repair procedure that incorporates the previous units. This procedure performs a complete diagnosis and then repairs any faults found.

2.4 Final Exams

Once the human or e-student has completed the entire curriculum, it is expected that they can handle a variation of each of the six problem types for which they have been trained. Each individual problem is graded as passing if and only if the simulated station is not working abnormally at the end of the last task. A student's score is the number of passing tests divided by number of tests total.

3 Phase III HD Curriculum

The Phase III version of the HD curriculum introduced novel learning challenges that had not been part of the Phase II diversity domain (training) curricula or the Phase II HD.These new challenges were incorporated into lessons that built directly on the Phase II curriculum, either by modifying existing lessons or adding additional rungs. These additional rungs required minor changes to the background knowledge as well as larger changes to the simulator and curriculum and final exam.

Some additional learning challenges only required changes to the curriculum encoded for the e-student. These changes, known as "relaxation trajectories," relaxed some of the formalisms required for the Phase II curriculum, such as not declaring the type of NIM, and removing excessive annotations of examples. These relaxation challenges, and their significance to the e-student, are highly technical in nature, and are not covered in this paper.

3.1 Simulator Changes

Updates to the Phase III ground station simulation were focused on three main areas. First, the orbit model used to simulate pointing at the satellite was updated to support a more realistic intrack error fault. Second, a real-time component was added to the simulator so that tasks could advance in real time and fault-based changes to the simulator state could happen at any point during task execution. Third, the fault definitions were extended to include modifiers to allow for additional versions of the existing faults.

3.2 Curriculum Changes

The curriculum changes involved two rung changes in Unit 3 (Repair Procedures) as well as the addition of two units.

Unit 3: Repair Procedures
First, the rung that teaches how to fix the intrack error was replaced. In the Phase II curriculum, an Auto-Bias correction function was added as a simple way for the student to fix any intrack error. This was replaced in Phase III with a more realistic and complicated procedure. In the new procedure, the student must perform a hill-climbing search to identify the proper adjustment. For example, the user might adjust the bias up to 8, then down to 4 and then back up to 6 to achieve the correct setting.

Second, the rung that teaches how to fix the Azimuth Error was changed to refer to the similar Elevation Error. That is, the Azimuth Error repair procedure is taught as being the same as the Elevation Error procedure, except that azimuth-related indicators and controls are used instead of elevation-related ones. This type of instruction is natural for human students.

Unit 6: Trigger Conditions
Three new rungs on trigger conditions place new requirements on the e-student but pose little challenge for human students. In Phase II, the student was asked to perform the over-arching *diagnose-and-repair* procedure after each task is executed. Instead, in Phase III the student is expected to promptly recognize when certain conditions occur and perform this procedure without being prompted.

Unit 7: Monitoring Time Bias
The two rungs in this unit teach the heuristic knowledge that once the intrack error has been fixed by manual adjustment, the user should watch the component indicators closely because it will likely need more adjustment in the future.

3.3 Final Exams

For purposes of evaluating human students in Phase III, automated grading was added to the simulator. The grader returns a single measure of whether the student solved a given problem before a fault-specific deadline – a yes or no answer. A yes answer means both the student fixed the fault, and at the end of the simulation there is a valid connection

from antenna to decommutator of at least the minimum allowed signal strength. This eased the process of grading for the larger number of students involved in Phase III testing.

One additional fault case was introduced in Phase III: no fault. When no fault was present, the student is not required to make any changes at all, as there is actually no problem for the student to fix.

4 Results

While the focus of this paper is on the hidden domain curriculum itself, we present a brief summary of the human and e-student Phase II and Phase III testing results with references to other works that include the complete testing methods, results, and discussion. An overview of the results is presented in Table 1. There is not a direct comparison between human and e-student results.

Table 1. Summary of final exam results.

	Phase II	Phase III
Human	91%	81%
e-student	100%	100%

In the Phase II curriculum, the 28 human students received a mean of 0% on the pre-test were able to successfully diagnose and repair 91% of the final exam problems [Grant et al., 2011]. In Phase III, 19 human students participated in the full curriculum, with additional students participating in other experiments where they received only some of the NIMs rather than all of them. The mean pre-test score was 0% and the mean post-test score was 81% [Grant et al., 2011]. These results do not include a final exam question on Unit 7, monitoring time bias.

For the e-student, the performance on the final exam in Phase II was 100%. In Phase III, the e-student was able to complete all of the tasks correctly. However, when a penalty is introduced for rungs that had to be injected, rather than learned, the performance is reduced to 96% (i.e., (rungs – injections) / rungs). That is, for 5 of the 7 final exam problems there was one rung where the e-student was unable to learn the concept and was supplied with the concept description (injected) so the e-student was able to continue onto other rungs. It is important to note that in Phase III the e-student was only tested with the relaxed formalisms and the additional Unit 6 on trigger conditions. The e-student was not tested on the updated repair procedures in Unit 3 or monitoring the time bias in Unit 7.

5 Conclusion

In this paper we describe the development of a "hidden domain" curriculum for the Bootstrapped Learning program. This automated curriculum was carefully designed to present appropriate learning challenges to the e-student, and to support instruction of university students in a reasonable amount of time. We provided details on the hidden domain curriculum in both Phase II and Phase III of the BL program. Finally we provided an overview of the results when the curriculum was taught to both human and e- students.

Acknowledgments

We want to thank our colleagues for their significant input throughout the curriculum development process: Manfred Bester, David DeAngelis, Robert Grant, Dan Luu, John Mohammed, Dewayne Perry, Benjamin Rode, Howard Reubenstein, Kathy Ryall, and Candy Sidner,

This material is based upon work supported by the United States Air Force Research Laboratory (AFRL) under Contract No. FA8650-07-C-7722. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

DISTRIBUTION STATEMENT A: Approved for Public Release, Distribution Unlimited

References

[Beal et al., 2009] J. Beal, P. Robertson, and R. Laddaga. Curricula and metrics to investigate human-like learning. In *Papers from the AAAI 2009 Spring Symposium: Technical Report SS-09-01*, Stanford, CA, 2009.

[Grant et al., 2011] Robert D. Grant, David DeAngelis, Dan Luu, Dewayne E. Perry and Kathy Ryall. Designing Human Benchmark Experiments for Testing Software Agents. In *Proceedings of EASE 2011*, Durham UK, April 2011.

[Ludwig et al., 2010] Jeremy Ludwig, John Mohammed, and Jim Ong. Developing an international space station curriculum for the bootstrapped learning program. In *Proceedings of the 2010 IEEE Aerospace Conference*, Big Sky, MT, March 2010.

[Mailler et al., 2009] Roger Mailler, Daniel Bryce, Jiaying Shen, and Ciaran O'reilly. MABLE: A Framework for Learning from Natural Instruction. In *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AA- MAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10– 15, 2009, Budapest, Hungary.

[Natarajan et al., 2010] Sriraam Natarajan, Gautam Kunapuli, David Page, Trevor Walker, Ciaran O'Reilly and Jude Shavlik. Learning from human teachers: Issues and challenges in bootstrap learning. In *AAMAS 2010 Workshop on Agents Learning Interactively from Human Teachers.* (www.ifaamas.org).

[Oblinger, 2006] Dan Oblinger. Bootstrapped Learning: Creating the electronic student that learns from natural instruction. Online, 2006. Available: http://www.darpa.mil/ipto/programs/bl/docs/AAAI_Briefing.pdf.