In Proceedings of the Argumentation in Multiagent Systems (ArgMAS) Workshop at Autonomous Agents and Multiagent Systems (AAMAS), Taipei, Taiwan, May 2011.

Toward the Application of Argumentation to Interactive Learning Systems

Elizabeth Sklar[1,2] and M. Q. Azhar[2]

[1] Dept of Computer & Information Science, Brooklyn College,
The City University of New York, 2900 Bedford Avenue, Brooklyn NY 11210, USA
[2] Dept of Computer Science, The Graduate Center,
The City University of New York, 365 Fifth Avenue, New York NY 10016, USA
sklar@sci.brooklyn.cuny.edu, mqazhar@gmail.com

Abstract. This paper explores the application of argumentation dialogues to an Interactive Learning System (ILS). The goal of an ILS is to provide an adaptive learning experience for a student within a particular domain, where the system adjusts dynamically as the student makes mistakes and learns from them. The system needs to be able to represent beliefs about the student's knowledge, and to update these beliefs as the student learns. The system also needs to have models of the domain and of an expert's actions within the domain, in order to compare and evaluate the student's actions. Finally, the system needs to provide appropriate feedback to the student, in such a way as to encourage learning. The work presented here describes a framework for such a system, built upon our earlier work on education dialogues.

1 Introduction

We explore the application of argumentation dialogues to an *Interactive Learning System (ILS)*. The goal of an ILS is to provide an adaptive learning experience for a student within a particular domain, where the system adjusts dynamically as the student makes mistakes and learns from them. The system needs to be able to represent beliefs about the student's knowledge, and to update these beliefs as the student learns. The system also needs to have models of the domain and of an expert's actions within the domain, in order to compare and evaluate the student's actions. Finally, the system needs to provide appropriate feedback to the student, in such a way as to encourage learning. The work presented here describes a framework for such a system.

Our model builds on earlier work in which we introduced the notion of an *education dialogue* [28]. Proposed for use in an interactive learning environment, an education dialogue is derived from previous work in the argumentation dialogue field [11, 20, 33]. Dialogues for education take place between two agents, each having specific roles: a *Tutor*, T, and a *Learner*, L. We focus on two types of interactions between these agents: $T \rightarrow L$ and $L \rightarrow T$, where the agent on the left side of the arrow initiates the dialogue, which is directed to the "target" agent on the right side of the arrow. Note that here we will not discuss $T \rightarrow T$

or $L \rightarrow L$ interactions, which, while possible in a general education dialogue, are not relevant for the specific instance discussed here.

Education dialogues are similar to *information seeking* dialogues [19, 33], but there are some key differences. When one agent asks another agent a question in an information seeking dialogue, the initiating agent does not know the answer and assumes that the target agent does. If the target agent does indeed know the answer, then she responds with the answer. However, in an education dialogue, there are reasons for the initiating agent to ask a question to which she already knows the answer and reasons for the target agent to not simply supply an answer she knows. Two such reasons are outlined below.

First, consider an education dialogue where the Tutor is the initiator, represented as $T \rightarrow L$. The Tutor actually does know the answer to the question she is posing. A good Tutor, pedagogically speaking, will ask a question that builds on the Learner's knowledge and coaxes him to learn; the answer will be something that the Tutor believes the Learner has the ability to find[3]. The Tutor is also refining her beliefs about the Learner's knowledge. Here, the Tutor is seeking information that is not the direct answer to the question, but rather she is seeking *meta-level knowledge* about the Learner—to see if the Learner knows the answer—instead of seeking the direct answer to her question (which, as stated, she already knows). Note that we make the assumption in the $T \rightarrow L$ interaction that the Learner will supply the answer if he knows it. There is a sizable literature from the educational psychology community on student motivation that explores reasons why a student might not answer a teacher's question correctly even if he knows the answer, but this avenue is outside the scope of the work discussed in this paper.

Second, consider an education dialogue where the Learner is the initiator, represented as $L \rightarrow T$. The Learner does not know the answer to the question she is posing (just like in a normal information seeking dialogue). If the Tutor knows the answer to the question, she may answer the question directly (as in an information seeking dialogue); or she may not provide the answer to the Learner, even though she knows it (unlike an information seeking dialogue). Since the Tutor's goal is to coax the learner to progress, she may decide to answer the Learner's question by providing more information about the answer, without providing the answer itself—to engage him in a thinking process that results in him learning.

The remainder of this paper is organized as follows. Section 2 discusses the specifics of education dialogues, reviewing some key components and introducing some new locutions. Section 3 briefly reviews the field of interactive learning systems, and focuses on highlighting components that are relevant to our framework. Section 4 describes our framework. Section 5 closes with a summary and discussion of future work.

[3] Note that for the remainder of this paper, we have arbitrarily chosen to use feminine pronouns to refer to the Tutor and masculine pronouns to refer to the Learner.

2 Education Dialogue Theory

The components of an education dialogue are as follows [22, 28]:

- Σ_i represents the *knowledge base*, or beliefs of each agent i. Thus, the Tutor's knowledge base is Σ_T and the Learner's knowledge base is Σ_L. The term Σ loosely refers to all the beliefs of an agent.

- An argument (S, p) is a pair, where p is the conclusion and S is the support for that conclusion. p is a logical consequence of S, and S is a minimal subset of Σ from which p can be inferred.

- $\mathcal{A}(\Sigma)$ is the set of all arguments that can be made from Σ.

- $\underline{S}(\Sigma)$ is the set of all acceptable arguments in Σ. Arguments that are acceptable are those that an agent has no reason to doubt: there are either no arguments that *undercut* them, or all the arguments that undercut them are themselves undercut by an acceptable argument.

- An agent's *commitment store*, $CS \in \Sigma$, refers to statements that have been made in the dialogue and which the agents are prepared to defend. CS_T refers to the Tutor's commitment store (statements the Tutor has made), and CS_L refers to that of the Learner. We can think of Σ as the agent's private knowledge base—all of the agent's beliefs—whereas CS is the agent's public knowledge base—all the beliefs that the agent has discussed in public (i.e., with other agents).

Parsons *et al.* [22] show how these simple elements can be used to construct common dialogues, such as information seeking dialogues.

In our earlier work [28], we introduced a new type of knowledge, which we call *meta-knowledge*. This is knowledge about the other agent(s) engaged in the dialogue, as perceived by each agent. We represent this meta-knowledge using Γ, which is a partition of Σ, in the same way that CS is. (Later, we will see that it is convenient to maintain these separate partitions of Σ.) We use the term $\Gamma_i(j)$ to refer to the meta-knowledge held by agent i about agent j. So, $\Gamma_T(L)$ refers to the Tutor's beliefs about the Learner's beliefs, i.e., what the Tutor believes is in the Learner's knowledge base, Σ_L. In addition, we use the $+$ modifier, as in $\Gamma_T(L+)$, to refer to the Tutor's beliefs about what the Learner can acquire. There is a vast literature on modeling the knowledge of learners, which is formally called *student modeling* (or *user modeling* in the more general sense) [21, 32]. Such models typically are designed for a specific domain, often in conjunction with the development of a particular tutoring system. Here we are not concerned with the precise details of individual student models, but rather use the concept abstractly in order to refer to the Tutor's meta-knowledge about the Learner—the Tutor's beliefs about what the Learner knows.

We can also use $\Gamma_L(T)$ to refer to the Learner's beliefs about the Tutor. This concept is useful, for example, for considering the Learner's motivation to learn

and his emotional state, both of which are discussed as important aspects for understanding human learners [16] and have been used in agent-based models of human behavior [30]. If the Learner does not believe that the Tutor knows the (correct) answers to questions about the Learner's domain, then he may be less motivated to progress when interacting with the Tutor. Take a real-world example: when students evaluate faculty members at the end of a term, it is common to ask the students to rate their professor's knowledge of the subject (the domain) covered in the course they just completed. Such a question assumes that the students form an opinion (acquire a model) of their professor's knowledge of (beliefs about) the domain. Elaboration on this aspect is beyond the scope of this paper, so here we will limit our discussion of Γ to refer only to the Tutor's beliefs about the Learner, $\Gamma_T(L)$.

2.1 Fundamental interactions

Below, we describe the fundamental steps in an interaction taking place between a Tutor and a Learner. The interaction is illustrated in Figure 1. Five fundamental steps and seven locutions are depicted. The goal is to arrive at the knowledge acquisition state in step 4. As noted below, some of the locutions are taken or derived from earlier work, primarily [1] and [23]. The *operational semantics* for each locution are detailed below (in boxes), in the order in which the locutions appear in the dialogue. The exchange is assumed to be a synchronized, turn-taking interaction that starts with the Tutor.

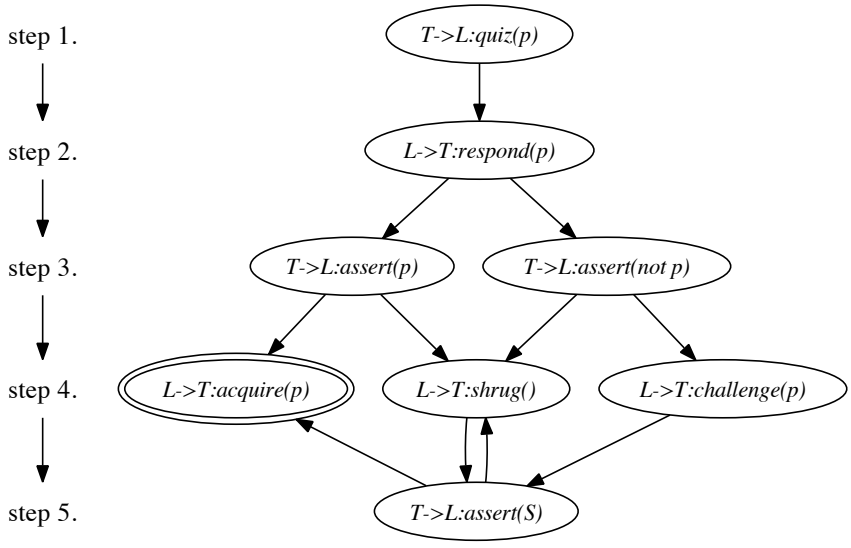


Fig. 1. Interaction sequence between a Tutor (T) and a Learner (L).

1. First, T poses a question to L about the verity of a proposition, p:

$$T \rightarrow L : quiz(p)$$

T is seeking to determine if the proposition p is in L's belief set. The Tutor knows the answer to the question, but does not know whether the Learner knows the answer. The goal of this dialogue is for the Tutor to determine if the Learner knows the answer.

quiz	
LOCUTION:	$T \rightarrow L : quiz(p)$
PRE-CONDITIONS:	1. $p \in \Sigma_T$
	2. $(S, p) \in \underline{S}(\Sigma_T)$
	3. $(S, p) \in \underline{S}(\Sigma_T \cup CS_L)$
	4. $p \in \Gamma_T(L+)$
	5. $(S, p) \in \underline{S}(\Gamma_T(L+))$
POST-CONDITIONS:	1. $CS_{T,i} = CS_{T,i-1} \cup \{p\}$ (update)
	2. $CS_{L,i} = CS_{L,i-1}$ (no change)
	3. $\Sigma_{T,i} = \Sigma_{T,i-1}$ (no change)
	4. $\Sigma_{L,i} = \Sigma_{L,i-1}$ (no change)
	5. $\Gamma_T(L)_i = \Gamma_T(L)_{i-1}$ (no change)

Note that this is semantically different from *question*(p) in an information seeking dialogue [1] because T already knows the answer to the quiz and so the purpose of the locution is to determine if L knows the answer. Although the format is similar to *question*(p), the pre and post conditions are sufficiently different that we have defined this new locution, *quiz*(p). A post-condition adds p to CS_T, even though $p \in \Sigma_T$, because this provides an explicit means for the Tutor to keep track of which propositions she has already discussed with the Learner. For convenient comparison, the operational semantics for *question* are listed in the Appendix at the end of this paper.

Also note the use of $\Gamma_T(L+)$ which represents the Tutor's belief that the Learner can find the answer to the question posed. The Tutor does not know for sure that $p \in \Gamma_T(L)$, but this notation permits the locution to be uttered and a reason for T believing that L can respond correctly.

2. Then, L responds to T's question:

$$L \rightarrow T : respond(p)$$

The Learner may or may not know the "right" answer—the correctness will be determined later in the dialogue by the Tutor. But in order to utter p, the Learner must possess some knowledge about p, either in its own knowledge base, Σ_L (meaning that the Learner has acquired p at some point), or in the Tutor's commitment store, CS_T (meaning that the Learner has not yet acquired p in its own knowledge base, Σ_L, but has the opportunity to do so, because it has heard p uttered by T at some earlier point in the dialogue and thus $p \in CS_T$).

respond	
LOCUTION:	$L \rightarrow T : respond(p)$
PRE-CONDITIONS:	1. $p \in (\Sigma_L \cup CS_T)$
POST-CONDITIONS:	1. $CS_{T,i} = CS_{T,i-1}$ (no change)
	2. $CS_{L,i} = CS_{L,i-1} \cup \{p\}$ (update)
	3. $\Sigma_{T,i} = \Sigma_{T,i-1}$ (no change)
	4. $\Sigma_{L,i} = \Sigma_{L,i-1}$ (no change)
	5. $\Gamma_T(L)_i = \Gamma_T(L)_{i-1}$ (no change)

The *respond* locution differs from the *assert* locution, discussed below, because it puts fewer requirements in the pre-conditions of the uttering agent. In order to be able to use $assert(p)$, the agent must believe p ($p \in \Sigma$) and must be able to support p ($(S, p) \in \Sigma$); whereas in order to be use $respond(p)$, the agent must either believe p or have heard the other agent state p: $p \in (\Sigma \cup CS)$.

3. The locution uttered by the Tutor in the next step depends on the correctness of the response given in the previous step.

   (a) If L has responded with the "correct" answer (i.e., T believes p), then T provides positive feedback, asserting p as described in [1]:

   $$T \rightarrow L : assert(p)$$

assert(p)	
LOCUTION:	$T \rightarrow L : assert(p)$
PRE-CONDITIONS:	1. $p \in \Sigma_T$
	2. $(S, p) \in \underline{S}(\Sigma_T \cup CS_L)$
POST-CONDITIONS:	1. $CS_{T,i} = CS_{T,i-1} \cup \{p\}$ (update)
	2. $CS_{L,i} = CS_{L,i-1}$ (no change)
	3. $\Sigma_{T,i} = \Sigma_{T,i-1}$ (no change)
	4. $\Sigma_{L,i} = \Sigma_{L,i-1}$ (no change)
	5. $\Gamma_T(L)_i = \Gamma_T(L)_{i-1}$ (no change)

   (b) If L has responded with the "incorrect" answer, (i.e., T believes $\neg p$), then T provides negative feedback by asserting $\neg p$:

   $$T \rightarrow L : assert(\neg p)$$

   The operational semantics of $assert(\neg p)$ are the same as above, by consistently substituting $\neg p$ for p.

4. The next step depends on the Tutor's response in the previous step, described above.

   (a) If L receives feedback from T that L understands, then L acknowledges that feedback and adds p (in the case of positive feedback) or $\neg p$ (in the

case of negative feedback) to its knowledge base[4]:

$$L \rightarrow T : acquire(p)$$

We leave discussion of how exactly the model of the Learner's knowledge base is updated to future work, and refer to [23] for the basis of that discussion.

acquire	
LOCUTION:	$L \rightarrow T : acquire(p)$
PRE-CONDITIONS:	1. $p \in (\Sigma_L \cup CS_T)^\dagger$
	2. $(S, p) \in \underline{S}(\Sigma_L \cup CS_T)$
POST-CONDITIONS:	1. $CS_{T,i} = CS_{T,i-1}$ (no change)
	2. $CS_{L,i} = CS_{L,i-1} \cup \{p\}$ (update)
	3. $\Sigma_{T,i} = \Sigma_{T,i-1}$ (no change)
	4. $\Sigma_{L,i} = \Sigma_{L,i-1} \cup \{p\}$ (update)
	5. $\Gamma_T(L)_i = \Gamma_T(L)_{i-1} \cup \{p\}$ (update)

†We note that it is not standard to allow an agent to utter p when p is not in its knowledge base (Σ), but this is not exactly the case here. In this case, the implementation of the locution includes processing steps in which the uttering agent (L) first adds p to Σ_L and then confirms that acquisition by uttering (essentially, reiterating) p.

(b) If L receives feedback from T that he does not understand, then L can pose a follow-up request for clarification. The appropriate locution is *challenge*(p), as outlined in [1], because the goal of L is to make T subsequently state her arguments in support of p:

$$L \rightarrow T : challenge(p)$$

challenge	
LOCUTION:	$L \rightarrow T : challenge(p)$
PRE-CONDITIONS:	1. $p \in CS_T$
POST-CONDITIONS:	1. $CS_{T,i} = CS_{T,i-1}$ (no change)
	2. $CS_{L,i} = CS_{L,i-1}$ (no change)
	3. $\Sigma_{T,i} = \Sigma_{T,i-1}$ (no change)
	4. $\Sigma_{L,i} = \Sigma_{L,i-1}$ (no change)
	5. $\Gamma_T(L)_i = \Gamma_T(L)_{i-1}$ (no change)

(c) If L receives feedback from T that he does not understand and L is so confused that he does not know what to say next, then he can shrug:

$$L \rightarrow T : shrug()$$

shrug	
LOCUTION:	$L \rightarrow T : shrug()$
PRE-CONDITIONS:	none
POST-CONDITIONS:	none

[4] For simplicity, we use p in the operational semantics description, but $\neg p$ could also be substituted, as long as the substitution was consistent.

This locution simply serves as a "no-op" (null operation) in order to be consistent with the turn-taking synchronized interaction in the implementation of our interactive learning framework (discussed in Section 4).

5. A final, optional, step occurs if the Learner does not understand the Tutor's feedback and has replied with a *shrug* or *challenge* locution in the previous step. In both cases, the Tutor responds by providing an explanation for p, using the $assert(S)$ locution described in [1]:

$$T \rightarrow L : assert(S)$$

assert(S)	
LOCUTION:	$T \rightarrow L : assert(S)$
PRE-CONDITIONS:	1. $p \in \Sigma_T$
	2. $(S, p) \in \underline{S}(\Sigma_T)$
POST-CONDITIONS:	1. $CS_{T,i} = CS_{T,i-1} \cup (S, p)$ (update)
	2. $CS_{L,i} = CS_{L,i-1}$ (no change)
	3. $\Sigma_{T,i} = \Sigma_{T,i-1}$ (no change)
	4. $\Sigma_{L,i} = \Sigma_{L,i-1}$ (no change)
	5. $\Gamma_T(L)_i = \Gamma_T(L)_{i-1}$ (no change)

3 Interactive Learning Systems

Intelligent Tutoring Systems (ITS) are a type of Interactive Learning System that provide users with opportunities to learn by interacting with a computer [26]. Unlike traditional computer-aided instruction, ITSs are not static, pre-programmed systems; rather, they adapt to students' responses. ITSs interject methodologies from artificial intelligence (AI) to manage that adaptivity, dynamically orchestrating users' learning experiences. An ITS uses a range of AI techniques to make decisions about which problem or information to present to a learner, and when and how to intervene if the learner makes mistakes.

Beck *et al.* [5] identify five major components in an ITS:

- The *domain model* contains the essential knowledge representation of the instructional domain. Both the pedagogical module and the student model (below) use the domain knowledge module to interpret a student's solution and track her skills.
- The *student model* records information about a student's performance with or misconception of the materials being taught. The idea is to build up a representation of a student's knowledge and skill set, updating this representation over time, as the student interacts with the system.
- The *pedagogical module*, or *tutor*, is the instructional, or teaching, component [27] which contains a set of rules about how to control and influence the student's learning process. The tutoring system uses this module to guide the student through the knowledge domain [29].

– The *expert model* contains knowledge about the cognitive structures and solution strategies underlying expertise in that particular domain. By using this model, the tutor can compare the student's solution with the expert's solution in order to figure out where learners have difficulties.
– The *communication module* provides the interface between the user and the tutor.

Classic ITS systems include the *LISP Tutor* [3, 8] for teaching the LISP programming language, and the *Andes* tutor [31] for teaching physics. Each is described briefly below.

The LISP Tutor [3, 8] incorporates *ACT**, a psychological theory of skill acquisition [2] and uses production rules and model tracing to model the tutor. Model tracing models errors that students make at each step on the basis of known misconceptions. By comparing the students' responses to the set of possible legal actions and the set of known wrong actions, the tutor is able to recognize whether the student is on a correct solution path, or appears be suffering from known misconceptions, or something unrecognizable. The student model in the LISP Tutor is partly descriptive and partly prescriptive. It is based on the authors' observations of students learning LISP and from the analysis of the required knowledge for LISP programming, as well as good programming styles. Procedural knowledge of how to write LISP code is modeled by a set of production rules.

Andes uses Bayesian networks for its student modeling component [7, 10, 18]. Every time the student selects a new problem, a Bayesian network is automatically generated. The structure of the the network is taken directly from a solution graph embedded in the system. The network contains five kinds of nodes:

– *Context-Rule nodes* model the ability to apply a rule in a specific problem-solving context in which it may be used. Each Context-Rule corresponds to a rule in Andes' ruled-based problem solver.
– *Fact nodes* represent the probability that the student knows a fact that is part of the problem solution.
– *Goal nodes* represents the probability that the student has been pursuing a goal that is part of the problem solution.
– *Strategy nodes* correspond to points where the student can choose among alternative plans to solve a problem.
– *Rule-Application nodes* represent the probability that the student has applied a piece of physics knowledge represented by a context-rule to derive a new fact or goal.

The Bayesian networks in Andes encode two kinds of knowledge: *domain-general knowledge*, which holds information about general concepts and procedures that define proficiency in Newtonian physics, and *task-specific knowledge*, which holds information related to student performance on a specific problem or example. Andes constitutes a probabilistic student model that provides long-term knowledge assessment, plan recognition, and prediction of students' actions during problem solving.

Over the last three decades, an extensive number of ITSs have been built using a range of techniques. Bayesian networks have been employed in multiple systems [6, 13]. Many have branched out to incorporate other techniques, such as object-oriented architectures (e.g., [34]). Various methodologies have been explored for emulating human best teaching practices, such as *coached program planning* [15], which helps students decompose problems. Some systems use natural language dialogues for interacting with students (e.g., [14]). An increasing number of systems take advantage of agent-based and multi-agent architectures [25]. Some incorporate intelligent interface components such as *pedagogical agents* [12]. However, to the best of our knowledge, no ITS system uses an argumentation-based framework or the education dialogues we have described above.

4 The ArgILS Framework

In this paper, we are concerned with the student model (the Learner), and the Tutor. Section 2 explained how to represent the Learner's knowledge (Σ_L), the expert's knowledge (Σ_T)[5], and the Tutor's knowledge about the Learner ($\Gamma_T(L)$); and provided an interaction structure for using that knowledge. This section introduces our *ArgILS* framework, which we have designed as a means for applying argumentation-based education dialogue theory to an interactive learning system. We describe our framework and ground it with an example.

The interaction sequence illustrated in Figure 1 and detailed in Section 2.1 outlines the fundamental series of steps in a theoretical education dialogue. This sequence is reasonable for interacting about *declarative* (factual) knowledge, where p can represent a fact and step 1 can be the Tutor asking the Learner if p is true. But the theory needs to be expanded in order to handle *procedural* knowledge. We need to provide a mechanism to communicate procedural information that cannot be expressed simply as a single proposition p. For example, the Tutor may ask the Learner how to execute a particular task, to which the Learner should be able to respond by uttering a series of propositions that all belong to a sequential procedure.

We represent a procedural sequence, $\overrightarrow{p}$ as:

$$\overrightarrow{p} = \{p_0, p_1, p_2, \ldots, p_{n-1}\}$$

Such a procedural sequence can be integrated into the interaction steps shown in Figure 1 in multiple ways. The first step, in which the Tutor puts forth a question to the Learner, remains essentially unchanged, with the substitution of $\overrightarrow{p}$ for p:

$$L \rightarrow T : quiz(\overrightarrow{p})$$

The second step, in which the Learner responds, however, will necessarily change.

[5] We make the assumption that the Tutor is the "expert".

Because the procedural knowledge is broken down into a number of pieces, there is a choice about redefining step 2 to:

$$L \rightarrow T : respond(\overrightarrow{p}) \tag{1}$$

where $\overrightarrow{p}$ represents all steps in the procedural sequence, or

$$L \rightarrow T : respond(\{p_i, p_{i+1}, \ldots, p_{j-1}, p_j\}) \tag{2}$$

where $\{p_i, p_{i+1}, \ldots, p_{j-1}, p_j\}$ represents some number of steps in the sequence, or

$$L \rightarrow T : respond(p_i) \tag{3}$$

where p_i represents one step in the procedural sequence.

One of the architecture decisions that arises in building an interactive learning system concerns *feedback*: when should the tutoring system provide help to the Learner? Equations 1 and 2 represent *delayed feedback*, where the Learner completes all or part of the task before receiving any feedback from the Tutor. Equation 3 represents *immediate feedback*, where the Learner completes only one step in the task before receiving feedback from the Tutor.

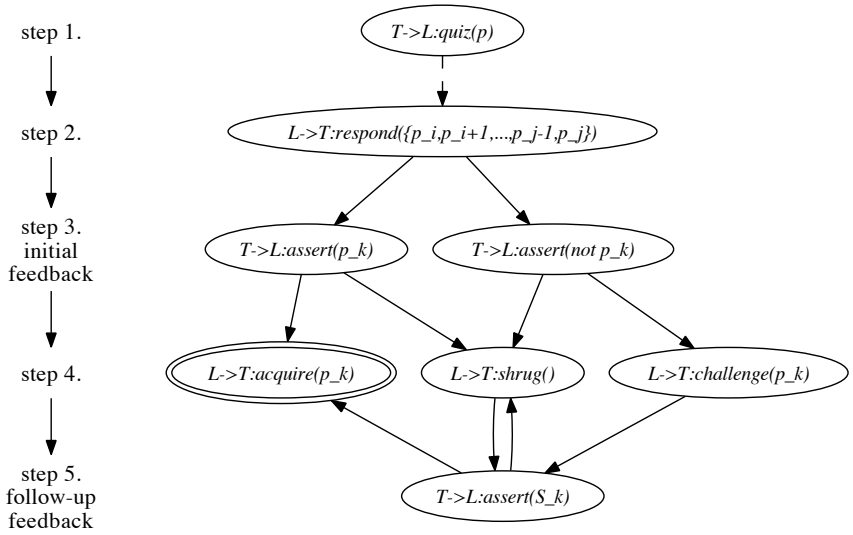


Fig. 2. Interaction sequence for *delayed feedback*.

Figure 2 illustrates an interaction sequence with delayed feedback. The first step is the same as in Figure 1, with the Tutor asking about the entire procedural sequence. The difference from Figure 1 lies in the second step, which is highlighted by the dashed line that leads from the first to the second step. In

the second step, the Learner responds with some number of propositions in the procedural sequence, as in equation 2. This can also correspond to equation 1, if $i = 0$ and $j = n - 1$ (i.e., equation 1 is just a specialized case of equation 2). The third step is the initial feedback step, where the Tutor comments on one of the propositions posited by the Learner, where $i \leq k \leq j$. The fourth, follow-up step and the fifth, follow-up feedback step proceed the same as in Figure 1, in response to the proposition p_k chosen by the Tutor in step 3. Note that the Tutor must decide which p_k to provide feedback for. Indeed, it is possible for the Tutor to comment on multiple p_k's; though for simplicity here, we only consider situations where the Tutor comments on one p_k at a time, and leave simultaneous commenting on multiple p_k's to future work.

Figure 3 illustrates an interaction sequence with immediate feedback. The picture is almost identical to that of Figure 1, with the difference being that the Tutor starts with $quiz(p)$, asking about the entire procedural sequence, and the Learner answers with a single step in the procedure: $respond(p_i)$. The dashed line from the first step to the second step highlights this difference.

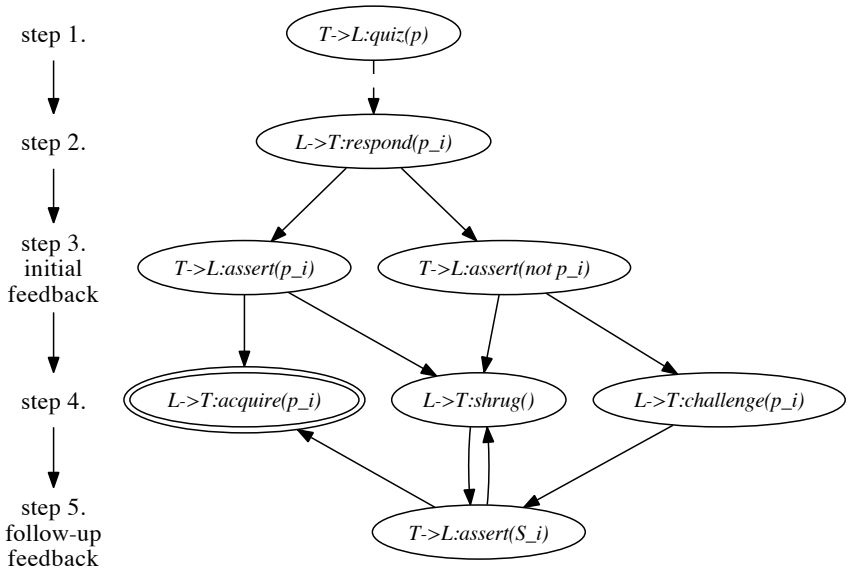
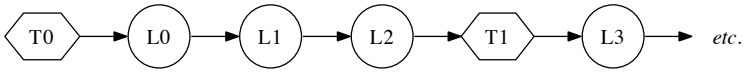


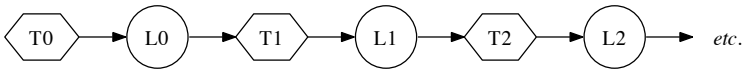
Fig. 3. Interaction sequence for *immediate feedback*.

Figures 2 and 3 illustrate the interactions over some portion of the procedural sequence. Unless the Learner provides the entire $\overrightarrow{p}$ and delayed feedback is employed and the Learner's response is entirely correct, some amount of iteration must occur before the Learner has received feedback on the entire procedural sequence. Figure 4 illustrates abstractly the differences in iteration patterns be-

tween delayed feedback and immediate feedback. With immediate feedack, every time the Learner makes an utterance, the Tutor replies immediately. With delayed feedback, the Tutor waits for the Learner to make several utterances, and then replies. The timing of the reply on the part of the Tutor in a delayed feedback system is another open area of research, and is something we will examine in future work. The important observation to make here is that we can model these differences in our ArgILS framework.



(a) delayed feedback



(b) immediate feedback

Fig. 4. Iterative sequences

Finally, we introduce one more locution, for use in iterative situations, as above, where the system is using immediate feedback—requiring that the Tutor respond immediately to every action on the part of the Learner. However, once the Learner acquires a proposition in the procedural sequence, he continues by positing the next step, without the Tutor reiterating the initial question. For just this case, in order to maintain the synchronized turn-taking in the iterative process, we introduce a "no-op" for the Tutor, which we call *nod*:

$$T \rightarrow L : nod()$$

nod
LOCUTION: $T \rightarrow L : nod()$
PRE-CONDITIONS: none
POST-CONDITIONS: none

4.1 An example interaction

Below we enumerate an example using our Human-Robot Tutoring System (HRTS) in which a Learner is trying to acquire knowledge about how to program a robot. Our HRTS is called RoboLite [4], and is based on the popular RoboLab [9, 24] programming interface originally designed for LEGO Mindstorms RCX robots [17].

In the first step, the Tutor utters:

$$T \rightarrow L : quiz(\overrightarrow{p}) \qquad \text{(step 1.)}$$

where $p =$ "How do you program a robot to go forward for 2 seconds and then stop?" Our system uses a graphical interface, where each command given to control the robot is represented as a building block icon. The expert's solution to the question is shown below:

p_0 p_1 p_2 p_3 p_4

In the second step, the Learner posits an icon. Let's say that the Learner starts with the correct icon, represented here by proposition p_0, so the Learner utters:

$$L \rightarrow T : respond(p_0) \qquad \text{(step 2.)}$$

In an immediate feedback system, the Tutor would immediately reply with positive feedback:

$$T \rightarrow L : assert(p_0) \qquad \text{(step 3a.)}$$

Since this is correct and the Learner's belief is confirmed, the Learner updates his knowledge base: $\Sigma_L = \Sigma_L \cup p_0$, and reiterates with:

$$L \rightarrow T : acquire(p_0) \qquad \text{(step 4a.)}$$

This is where the null operation is needed for the Tutor, to maintain the turn-taking pattern, but without reiterating any propositions unnecessarily or introducing anything new. Thus, the Tutor indicates that the Learner should proceed by uttering:

$$T \rightarrow L : nod()$$

Now the Learner adds another icon. Let's say he makes a mistake and enters p_4:

$$L \rightarrow T : respond(p_4) \qquad \text{(step 2.)}$$

so his partial solution would look like this:

p_0 p_4

Again, in an immediate feedback system, the Tutor would reply immediately. The Tutor compares the Learner's sequence, $\{p_0, p_4\}$, with the expert sequence, $\{p_0, p_1, \ldots\}$, and detects an anomaly with the second proposition in the sequence. So this time, the Tutor comments with negative feedback:

$$T \rightarrow L : assert(\neg p_4) \qquad \text{(step 3b.)}$$

The Learner does not understand why his sequence is incorrect, so he requests clarification by uttering:

$$L \rightarrow T : challenge(\neg p_4) \qquad \text{(step 4b.)}$$

whereby the Tutor responds by providing the reasons why p_4 is the incorrect proposition in the sequence:

$$T \rightarrow L : assert((S, \neg p_4)) \qquad \text{(step 5.)}$$

An alternative to the Tutor providing a negative assertion (as in step 3b, above) is for the Tutor to provide the Learner with the right answer by uttering:

$$T \rightarrow L : assert(p_1) \qquad \text{(step 3a.)}$$

If the Learner does not understand, then he would again ask for clarification:

$$L \rightarrow T : challenge(p_1) \qquad \text{(step 4b.)}$$

and the Tutor would supply the reasons why p_1 is the correct proposition in the sequence:

$$T \rightarrow L : assert((S, p_1)) \qquad \text{(step 5.)}$$

In a delayed feedback system, the Tutor would wait until the Learner had entered several icons before commenting. The questions of when to respond and how to respond are areas of future research to be addressed in the development of our implemented system. The ArgILS provides a solid framework in which to model the possibilities.

5 Summary

We have described an extended education dialogue system, expanding on our earlier work and that of others in the argumentation dialogue community. We have introduced ArgILS, our general framework for an interactive learning system in which interactions between a Tutor and a Learner can be modeled. An example was provided, demonstrating the use of ArgILS in the development of our work-in-progress Human-Robot Tutoring System. Multiple avenues of future work have been identified, such as the Tutor's choice of which proposition to comment on in a delayed feedback system for procedural knowledge and when to provide comments in a delayed feedback system.

Appendix

Below are the operational semantics of the *question* locution, adapted from [1]. A question is posed when the initiating agent, T in the description below, asks a question of another agent, L in the description below. In the case of a question, it is assumed that the asking agent does not know the answer to the question, p in the description below. In addition, the asking agent does not know whether the target agent knows the answer or not. (This is revealed in the choice of response locution subsequently executed by the target agent.)

question	
LOCUTION:	$T \rightarrow L : question(p)$
PRE-CONDITIONS:	1. $(S, p) \notin \underline{S}(\Sigma_T)$
	2. $(S, \neg p) \notin \underline{S}(\Sigma_T)$
POST-CONDITIONS:	1. $CS_{T,i} = CS_{T,i-1}$ (no change)
	2. $CS_{L,i} = CS_{L,i-1}$ (no change)
	3. $\Sigma_{T,i} = \Sigma_{T,i-1}$ (no change)
	4. $\Sigma_{L,i} = \Sigma_{L,i-1}$ (no change)

Acknowledgments

We would like to thank Simon Parsons, for his collaboration on our original work on education dialogues. We would also like to extend our gratitude to the reviewers of this paper, who provided many thought-provoking comments that will help us shape some of the next steps in the development of our framework.

References

1. Amgoud, L., Maudet, N., Parsons, S.: Modelling dialogues using argumentation. In: Proceedings of the 4th Conference on Multi-Agent Systems. Boston (2000)
2. Anderson, J.R.: The Architecture of Cognition. Harvard University Press (1983)
3. Anderson, J.R., Skarecki, E.: The automated tutoring of introductory programming. Communications of the ACM 29(9), 842–849 (September 1986)
4. Azhar, M.Q., Goldman, R., Sklar, E.I.: An agent-oriented behavior-based interface framework for educational robotics. In: Agent-Based Systems for Human Learning (ABSHL) Workshop at Autonomous Agents and MultiAgent Systems (AAMAS) (2006)
5. Beck, J., Stern, M., Haugsjaa, E.: Applications of AI in Education. Crossroads 3(1), 11–15 (1996)
6. Beck, J.E., m. Chang, K., Mostow, J., Corbett, A.: Does help help? introducing the bayesian evaluation and assessment methodology. In: 9th International Conference on Intelligent Tutoring Systems. pp. 383–394 (June 2008)
7. Conati, C., Gertner, A., VanLehn, K.: Using Bayesian Networks to Manage Uncertainty in Student Modeling. User Modeling and User-Adapted Interaction 12(4) (2002)
8. Corbett, A.T., Anderson, J.R.: The LISP Intelligent Tutoring System: Research in skill acquisition, pp. 73–109. Lawrence Erlbaum, Hillsdale, NJ, USA (1992)
9. Erwin, B., Cyr, M., Rogers, C.B.: LEGO Engineer and ROBOLAB: Teaching Engineering with LabVIEW from Kindergarten to Graduate School. International Journal of Engineering Education 16(3) (2000)
10. Gertner, A.S., Conati, C., VanLehn, K.: Procedural help in Andes: Generating hints using a Bayesian network student model. In: Proceedings of the National Conference on Artificial Intelligence (AAAI). pp. 106–111. AAAI Press (1998)
11. Girle, R.: Commands in Dialogue Logic. In: Practical Reasoning: Proceedings of the First International Conference on Formal and Applied Practical Reasoning (FAPR). pp. 246–260. Lecture Notes in Artificial Intelligence 1085, Springer, Berlin, Germany (1996)

12. Johnson, W.L., Rickel, J.W., Lester, J.C.: Animated pedagogical agents: Face-to-face interaction in interactive learning environments. International Journal of Artificial Intelligence in Education 11 (2000)
13. Kasurinen, J., Nikula, U.: Estimating programming knowledge with bayesian knowledge tracing. In: Proceedings of the 14th annual ACM SIGCSE conference on Innovation and Technology in Computer Science Education (ITiCSE). pp. 313–317. ACM, New York, NY, USA (2009)
14. Lane, H.C., VanLehn, K.: A dialogue-based tutoring system for beginning programming. In: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS). pp. 449–454. American Association for Artificial Intelligence Press (2004)
15. Lane, H., VanLehn, K.: Coached program planning: dialogue-based support for novice program design. In: Proceedings of the 34th SIGCSE technical symposium on Computer science education. pp. 148–152. ACM (2003)
16. Lazarus, R.S.: Cognition and Motivation in Emotion. American Psychologist 45(4), 352–367 (1991)
17. LEGO Mindstorms. http://www.legomindstorms.com/
18. Martin, J., Vanlehn, K.: Student assessment using bayesian nets. International Journal of Human-Computer Studies 42, 575–591 (1995)
19. McBurney, P., Parsons, S.: Agent ludens: Games for agent dialogues. In: Proceedings of the AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents. Stanford, CA, USA (2001)
20. McBurney, P., Parsons, S.: Chance discovery using dialectical argumentation. In: Proceedings of the Workshop on Chance Discovery, Fifteenth Annual Conference of the Japanese Society for Artificial Intelligence. Matsue, Japan (2001)
21. McCalla, G.I., Greer, J.E.: Granularity-based reasoning and belief revision in student models. In: Student Models: The Key to Individualized Educational Systems. pp. 39–62. Springer Verlag, New York (1994)
22. Parsons, S., Wooldridge, M., Amgoud, L.: Properties and complexity of formal inter-agent dialogues. Journal of Logic and Computation 13(3), 347–376 (2003)
23. Parsons, S., Sklar, E.: How agents alter their beliefs after an argumentation-based dialogue. In: Proceedings of the Workshop on Argumentation in Multiagent Systems (ArgMAS) at Autonomous Agents and MultiAgent Systems (AAMAS) (2005)
24. Robolab. http://www.ceeo.tufts.edu/robolabatceeo/
25. Sklar, E.I., Richards, D.: Agent-based systems for human learners. Knowledge Engineering Review 25(2), 111–135 (June 2010)
26. Sklar, E.: CEL: A Framework for Enabling an Internet Learning Community. Ph.D. thesis, Department of Computer Science, Brandeis University, Waltham, MA, USA (2000)
27. Sklar, E., Davies, M.: Multiagent simulation of learning environments. In: Fourth International Conference on Autonomous Agents and Multi Agent Systems (AAMAS) (2005)
28. Sklar, E., Parsons, S.: Towards the Application of Argumentation-based Dialogues for Education. In: Proceedings of the Third International Conference of Autonomous Agents and Multi Agent Systems (AAMAS). pp. 1420–1421 (2004)
29. Sklar, E., Richards, D.: The use of agents in human learning systems. In: Fifth International Conference on Autonomous Agents and Multi Agent Systems (AAMAS) (2006)
30. Spoelstra, M., Sklar, E.: Using simulation to model and understand group learning. Agent Based Systems for Human Learning, International Transactions on Systems Science and Applications 4(1) (2008)

31. VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M.: The Andes Physics Tutoring System: Lessons Learned. International Journal of Artificial Intelligence and Education 15(3) (2005)
32. VanLehn, K., Niu, Z., Slier, S., Gertner, A.: Student modeling from conventional test data: A bayesian approach without priors. In: Proceedings of the 4th Intelligent Tutoring Systems Conference (ITS). pp. 434–443 (1998)
33. Walton, D.N., Krabbe, E.C.W.: Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. State University of New York Press, Albany, NY, USA (1995)
34. Wei, F., Moritz, S.H., Parvez, S.M., Blank, G.D.: A student model for object-oriented design and programming. Journal of Computing Sciences in Colleges 20(5), 260–273 (2005)