The Architecture of Why2-Atlas: A Coach for Qualitative Physics Essay Writing

Kurt VanLehn, Pamela W. Jordan, Carolyn P. Rosé, Dumisizwe Bhembe, Michael Böttner, Andy Gaydos, Maxim Makatchev, Umarani Pappuswamy, Michael Ringenberg, Antonio Roque, Stephanie Siler, and Ramesh Srivastava

LRDC, University of Pittsburgh, Pittsburgh, PA 15260
vanlehn@cs.pitt.edu

Abstract. The Why2-Atlas system teaches qualitative physics by having students write paragraph-long explanations of simple mechanical phenomena. The tutor uses deep syntactic analysis and abductive theorem proving to convert the student's essay to a proof. The proof formalizes not only what was said, but the likely beliefs behind what was said. This allows the tutor to uncover misconceptions as well as to detect missing correct parts of the explanation. If the tutor finds such a flaw in the essay, it conducts a dialogue intended to remedy the missing or misconceived beliefs, then asks the student to correct the essay. It often takes several iterations of essay correction and dialogue to get the student to produce an acceptable explanation. Pilot subjects have been run, and an evaluation is in progress. After explaining the research questions that the system addresses, the bulk of the paper describes the system's architecture and operation.

1 Objectives

The Why2 project has three objectives. The first is to build and evaluate qualitative physics tutors where all student communication is via natural language text. In particular, we will compare their performance to text-based expository instruction and human tutoring. Text-based natural language (NL) technology has improved significantly but is still far from perfect. Are the inevitable disfluencies so confusing that they significantly retard learning, or are they no worse than the disfluencies of human-to-human text-based tutoring?

The second objective is to compare several different NL processing techniques. In particular, we are collaborating with the AutoTutor Research Group [1, 2] who are building a version of Why2 using latent semantic analysis, a statistical technique. Our version of Why2 is called Why2-Atlas; their version is called Why2-AutoTutor. Why2-Atlas is based on deep syntactic analysis and compositional semantics. It is more difficult to build but may yield better performance.

The third objective is to develop authoring tools that facilitate development of NL-based tutoring systems. Whereas a typical tutoring system's size is roughly determined by the amount of material it covers, an NL-based tutor's size is also a function of the diversity of the words and linguistic structures that students use.

Since this is a large effort, we chose a worthy pedagogical goal: qualitative physics understanding. It is well known that college physics students are often unable to construct acceptable qualitative explanations for simple physics questions, such as "what happens when two balls of different masses fall from the same height?" Sometimes they know what happens (the balls hit the ground at the same time) but they cannot explain *why* it occurs. Decades of experimental instruction have produced only limited progress toward universal qualitative understanding [3]. Even students with top grades in their classes get low scores on standardized measures of qualitative understanding, such as the Force Concepts Inventory [4].

The Why2-Atlas user interface is remarkably simple. There are three windows: the problem window, the essay window and the chat window. The problem window merely states a question, such as "Suppose you are running in a straight line at constant speed. You throw a pumpkin straight up. Where will it land? Explain." Students type their explanation into the essay window, then click on a submit button. The student and the tutor take turns typing in the chat window. For instance, one student's explanation was, "It would seem as though the ball would fall back into the man's hands at the end of T, but I do not think that this will happen…after the ball is thrown there is a lack of a force which results in the decrease of horizontal velocity." When this is entered in the essay window, Why2-Atlas analyzes it, which involves determining what "the ball" and "the man" refer to, among other things. The tutor determines that the student has a common misconception, which is that a moving object always slows down unless there is a force to propel it along. Switching to the chat window, the tutor engages the student in a dialogue, such as the one shown in Table 1. At its end, the tutor suggests that the student change the essay. If all goes well, the student will remove the misconception from the explanation, in which case the tutor can address any flaws that remain. If the student does not do so, then the tutor will either try a different dialogue.

Why2-Atlas is composed of several modules: A sentence-level understander (SLU), a discourse-level understander (DLU), a tutorial strategist and a dialogue engine. These four modules are controlled by a discourse manager. As the student enters an essay, the SLU parses it and produces a set of propositions expressed in a version of first-order logic. The discourse-level understander (DLU) assimilates them into a explanation. If a proposition is new, then the DLU augments the explanation with inferences that explain (prove abductively) the proposition. Next, the tutorial strategist analyzes the augmented explanation's completeness and correctness. If the explanation used rules that represent misconceptions (e.g., "If there is no force to propel an object along, it slows down."), then the tutorial strategist creates tutorial goals to remedy those misconceptions. If the explanation does not explicitly mention one of the points that the project physicists have indicated must be present in any acceptable essay, then the strategist creates a tutorial goal of eliciting that mandatory point. Next, the strategist uses heuristics to select the highest priority tutorial goal and attempts to achieve it by calling the dialogue engine to conduct a knowledge construction dialogue (KCD). If the student fails to remove the flaw after one or more KCDs, the system tells the student how to do so.

The remaining sections describe each of the modules in more detail and the relationship of this work to earlier work.

Table 1. A Knowledge Construction Dialogue (KCD)

- Tutor: OK, let's consider the relationship between force and acceleration in a different context. A hockey puck is resting on the rough ice surface of a skating rink (a hockey team has just finished practicing on it). You push with a force, F, during the time from t1 to t2. Assume that the only force acting horizontally on the puck is friction due to the ice surface (there is no air resistance). What will happen to the velocity of the puck after your push?
- Student: it will get slower
- Tutor: Good. A zamboni smoothes the ice so that the friction caused by the roughness of the ice surface is reduced. You push the same hockey puck with the same force as before (and for the same amount of time). How does the change in velocity of the hockey puck after this push compare to its change in velocity following the initial push?
- Student: slows more slowly
- Tutor: Correct. Now assume there exists a machine that can make the ice surface perfectly smooth, so that the friction caused by the ice surface is zero. Again assume the force of air resistance is zero. If you were to push a puck on this surface, what would happen to the velocity of the puck following the push?
- Student: nothing
- Tutor: Right. In general, according to Newton's first law, if there are no forces acting on a body, what happens to the body's velocity?
- Student: not a darn thing
- Tutor: Okay. In general, these and similar observations are summarized in Newton's first law, which states that if there are no forces acting on a body, its velocity will remain constant (or its acceleration will be zero)

2 The Sentence-Level Understander

The sentence-level understander (SLU) converts each sentence in the student's essay into a set of propositions. The propositions are expressed in a version of first-order logic. For instance, consider the following sentence: "Should the arrow have been drawn to point down?" Roughly speaking, this should be converted to:

$$\exists e \in \text{events}, \exists v \in \text{vectors}, \exists s\ \text{draw}(e, s, v) \& \text{tense}(e, \text{past}) \& \text{mood}(e, \text{interrog}) \& \text{direction}(v, \text{down})$$

This is only an approximation of the real output. It illustrates the challenge of converting words into the appropriate domain-specific predicates. Notice how "arrow" has been converted to "vector." Another challenge is to interlink the predicate arguments correctly: Notice how the variable that denotes "the arrow" appears in both "draw" and "direction." Other aspects of the meaning of the utterance are not yet processed. For instance, "the arrow" probably refers to some specific vector mentioned earlier in the dialog, but the SLU makes no attempt to resolve that reference.

The SLU is composed of a lexical preprocessor, a parser, a repair module (which together comprise a package called CARMEL) and a statistical analyzer [5, 6]. As the student types in a sentence, the text is sent to the lexical preprocessing module, which looks up the words in a lexicon, doing spelling correction as necessary. It will also strip off prefixes and suffixes in order to uncover the root forms of the words. The resulting graph of root word forms is sent to the next module, the parser.

LCFlex [5, 7], is a flexible left-corner parser that does a deep syntactic analysis of each sentence. For instance, for the illustration sentence, the parser determines that "the vector" is both the deep object of "draw" and the deep subject of "point." As it builds a syntactic analysis, the parser builds the logical form as well. This allows it to check that the predicates are assigned arguments of the right type.

LCFlex copes with ungrammatical input by skipping words, inserting missing categories and relaxing grammatical constraints as necessary in order to parse the sentence. For instance, "Should the arrow has been drawn point down" would parse. When the parser produces too many analyses of the sentence, it uses statistical information on the frequency of word roots and grammatical analyses to determine the most probable parse.

If the parser cannot produce any complete analysis of the sentence, then its fragmentary analysis is passed to the repair module. The fragments can be viewed as domain-specific predicates that are looking for argument fillers, and domain-specific typed variables that are looking for arguments to fill. A genetic search is used to build a complete analysis from these fragments.

If the repair module fails, then the symbolic approach is abandoned, and the SLU's fourth module takes over. It uses text classification techniques. Given a statistical language model that indicates how strongly each word is associated with domain classifications (e.g., misconceptions), it computes the most likely classification of each sentence using either a naïve-Bayesian approach [8] or LSA.

Besides this statistical language model, the SLU requires three other large knowledge sources: a meaning-representation language definition, a grammar and a lexicon. The meaning-representation language definition specifies the domain predicates and types that can occur in the logical form output by the SLU. The definition of a predicate specifies type restrictions on its arguments. For instance, the definition of the "move" predicate specifies that its "agent" argument be an instance of the "concrete entity" type.

The grammar is a unification-augmented context-free grammar based on both Functional Grammar [9] and Lexical Functional Grammar [10]. Its main job is to assign deep syntactic roles to phrases in the sentence. For instance, in both the active sentence "I drew the vector" and the passive sentence "The vector was drawn," the noun phrase "the vector" is assigned to the same deep syntactic role (argument) of the verb "draw."

The lexicon defines word roots. For each one, it specifies their syntactic categorization, the possible semantic predicates or types, and the mapping from syntactic roles to semantic roles. For words like verbs that can have arguments, the lexical entry also specifies information about the syntactic and semantic argument restrictions. Words with multiple senses have multiple lexical entries. For instance, "draw" has 7 entries. The lexicon was built by adding semantic information and idioms to COMLEX [11].

Parts of the SLU (e.g., LCFlex) have been evaluated separately [5, 7], and some are being used by other projects [12, 13].

3 Discourse-Level Understanding

The discourse-level understander (DLU) receives logical forms and outputs a proof. Topologically, the proof is a forest of interwoven trees. The leaves are facts given in the problem statement or assumptions made while the proof is being constructed. The roots (conclusions) are student propositions. Other student propositions may occur in the middle of the forest. As an illustration, consider the following question and student essay:

> Q: Suppose you are in a free-falling elevator and you hold your keys motionless in front of your face and then let go. What will happen to them? Explain.
>
> A: The keys will fall parellel to the persons face because of the constant acceleration caused by gravity but later the keys may go over your head because the mass of the keys are less.

Although this essay has a misspelling, missing punctuation and grammatical errors, the SLU parses it and sends four propositions to the DLU. The first proposition corresponds to the student statement that the keys will fall parallel to the person’s face. This conclusion is correct and becomes a root in the proof. The second proposition, that gravitational acceleration is constant, corresponds to an interior node in this proof. The third proposition, that the keys go over the person’s head, is based on a common misconception, that heavier objects fall faster. It becomes a root in the proof. The last proposition, that the mass of the keys is less, corresponds to a node in the interior of the proof of the third proposition.

Proofs are constructed by Tacitus-lite+ [14], which is an extension of Tacitus [15]. Its knowledge base is a set of Horn clauses that represent both correct physics beliefs, such as “If a vector is vertical, its horizontal component is zero,” and incorrect beliefs, such as “more massive objects have larger vertical accelerations.” The goals in the body (the antecedents) of a rule each have a cost. Tacitus-lite+ can choose to assume the goal rather than prove it, in which case its cost is added to the overall cost of the proof. Tacitus-lite+ prefers proofs with the lowest costs. This mechanism, which is a form of abduction, is necessary to allow Tacitus-lite+ to “prove” false student statements.

When a goal that has unbound variables is proved, the variables often become bound to constants, terms or other variables. When Tacitus-lite+ assumes a goal, it can also bind unbound variables. To reduce the combinatorics when this occurs, all arguments in predicates and terms have types associated with them. The types constrain what existing constants, terms and variables may be bound to the unbound arguments of the goal that is being assumed. If a variable is used in several goals, it accumulates constraints from all of them.

Because variables accumulate constraints, the process of resolving referring expressions is interwoven with the construction of the proof. For instance, the student statement, “the mass of the keys is less,” becomes a proposition similar to less(mass(keys1), X) where keys1 is a constant denoting the keys, mass(keys1) is a compound term denoting their mass, and X is a variable. That is, the student did not say what the mass was less than, so the DLU must resolve the implicit reference. Via accumulation of constraints and some discourse heuristics based on centering [16], the DLU ends up binding X to mass(person1). Sometimes objects must be created in order to refer to them. For instance, the temporal reference implicit in the student's statement, "later the keys may go over your head," requires dividing the fall of the keys into two time intervals, and stating that the keys are parallel to the person's face during the first interval but above the person's head during the second interval.

Creating time intervals in order to refer to them is just one of many complexities that must be handled when converting language to proofs. We have only tried to solve those that are necessary for our purposes. For instance, the modal "may" is ignored in "later the keys may go over your head."

4 The Tutorial Strategist

Once a proof has been constructed, the tutorial strategist analyzes it to find flaws. Each flaw is associated with patterns that match occurrences of the flaw in the proof. In the proof mentioned earlier, there are several flaws. The main one is that the misconception "heavier objects fall faster" has been used. In addition, several points that physicists consider mandatory for a sufficiently complete explanation have been implied but not stated. For instance, one is "the keys' acceleration and the person's acceleration are the same." Each detected flaw is queued as a tutorial goal to remedy the flaw.

Once tutorial goals have been queued, the strategist picks the highest priority goal from the queue. The priorities are: fix misconceptions before anything else, then fix self-contradictions, errors and incorrect assumptions, and lastly elicit missing mandatory points. Since most essays are missing several mandatory points, the choice of which one to tutor first is determined by a hand-authored list for each problem. This insures that the points will be elicited in a natural order.

5 The Dialogue Engine

The tutorial goals for remedying a misconception are associated with a specific *remediation* KCD, and those for eliciting a mandatory point are associated with a specific *elicitation* KCD. If one of these goals is chosen, then the dialogue manager, APE, is called with the name of the KCD.

KCDs are managed by finite state networks whose nodes are questions to the students. The links exiting a node correspond to expected responses to the question. The questions are written to invite short responses from the students so that simple techniques can be used to match the expected responses to the actual ones. For instance, Table 1 shows one KCD.

In order to determine which expected response is the best match to the student's answer, APE calls LCFlex with a simple semantic grammar. Each question has its own semantic grammar, although the grammar may share rules with other questions' grammars. The root categories in the grammar correspond to the expected responses. For instance, if the expected responses are "down" and "up," then the semantic grammar would have two rules such as "Ques32_down_resp => down_cat" and "Ques32_up_resp => up_cat" where down_cat and up_cat are categories used in several semantic grammars and are reduced by rules such as "down_cat => 'down'," "down_cat => 'downwards'," "down_cat => 'towards earth'," etc. Because LCFlex can skip words, it can find certain key words or phrases in the student's response even if they are surrounded by extra words, as in "Is it downwards?"

When the student has finished the KCD, the discourse manager inserts a segue, such as "Given what you've learned, please change your essay." The student should edit the essay, and resubmit it. When students fail to correct the essay, the same flaw will be detected and will probably be selected again for fixing. If the tutorial goal is important enough that multiple KCDs exist for remedying it, then a second KCD will be tried. If the discourse manager runs out of KCDs for fixing a flaw, then it gently suggests an explicit change to the essay. For example, if one of the required points is missing, the tutor says, "It could be that you have what I'm looking for in mind, but I'm just not able to understand what you're saying. Let me show you what I'd say was the point that should be covered in your essay: 'After their release, the only force acting on the keys is the downward force of earth's gravity.'"

6 Authoring Tools

The major knowledge sources in Why2-Atlas are the Tacitus-lite+ rules, the KCDs, the semantic grammars used in the KCDs, the syntactic grammar and the lexicon. We are working on tools for authoring all these knowledge sources. However, the only tools that are currently in routine use are the KCD editor and DAIENU, the semantic grammar authoring system.

The KCD editor is a graphically oriented editor designed for non-programmers [17, 18]. Typically the author starts by creating a main line of reasoning consisting of alternating questions and correct responses. Next the author enters expected incorrect answers for each question. The author then defines subsidiary KCDs, some of which may also be remediation KCDs, for the incorrect answers. It is typical to call the same KCDs from many different locations. As a result, KCDs can become so complicated that it would be difficult to navigate their links without the aid of the KCD editor.

The second tool, DAIENU, is used to create the semantic grammars required by the KCDs [19]. There are several hundred KCDs in the current version of Why2-Atlas, and each has several questions, so several hundred semantic grammars are needed. DAIENU begins by collecting all the expected student responses in the KCDs. Each is a string of expected student words, entered by the author, and a category standing for that particular expected response. The job of DAIENU is to create intermediate categories, similar to the down_cat and up_cat mentioned earlier, that will be used in the multiple semantic grammars. DAIENU suggests clusters of strings, based on shared words, to the author. The author can also add or subtract strings from the initial set. Together they come up with a set of strings that occurred in the KCD responses and have similar meaning, according to the author. DAIENU then finds commonalities between the sets such that it can construct a compact set of general grammar rules that will correctly classify the members of each set. When it has produced a candidate set of grammars, it generates novel strings labeled by the grammars, and presents them to the author. The author corrects their classifications, and the whole grammar induction process repeats.

7 Related Work

Development of Why2-Atlas is possible only because we borrowed many ideas from earlier efforts. This section highlights some of the similarities between Why2-Atlas and other NL-based tutoring systems.

Perhaps the first fully operational NL-based tutor was CIRCSIM-tutor [20]. It used information-extraction techniques and short-answer questions in order to conduct a dialogue about the student's qualitative analysis of a cardiophysiological feedback system. We also use short answer questions and word-based analysis of student responses in order to conduct a robust, fluent dialogue. In fact, the two tutors use nearly the same dialogue engine [21]. However, the CIRCSIM-tutor students express their cardiophysiological analysis by putting +/-/0 marks in a table, whereas our students enter an NL essay. Most of the current complexity in Why2-Atlas is devoted to analysis of this essay.

Another early NL-based tutor is AutoTutor [2]. It opens the dialogue by simply asking the student a question, such as "What happens when you boot up a computer?" It has a list of mandatory points that it wants the student to explicitly articulate, and will prompt the student more and more pointedly until the student says (types) them clearly. It also has a set of misconceptions that it anticipates the student will enter, and has remedial messages to deal with them. We use the same basic idea for analyzing the student essays: a set of mandatory points and a set of misconceptions. The main difference is that AutoTutor uses a statistical technique (LSA) to determine if a point has been mentioned, whereas Why2-Atlas uses symbolic analyses.

Whereas CIRCSIM-tutor and AutoTutor use shallow, word-based analyses of student text, Aleven, Popescu and Koedinger [12] are building a tutor that uses deep analyses of student explanations. Their tutor asks students simple geometry questions, such as whether two angles are the same. Students must provide both an answer and a NL explanation, such as "The sum of complementary angles is 180 degrees." Students typically do not provide such precise, complete explanations on their first attempt, so the tutor must prompt them more and more pointedly to add precision to their justification. Shallow, word-based analyses of these explanations would clearly not suffice, so the tutor uses the LCFlex parser [5, 7], a deep syntactic grammar, the Loom symbolic classifier, and a geometry ontology to analyze the students' explanations. For the same reasons, we also use the LCFlex parser and a deep syntactic grammar. However, our students' explanations often contain several clauses, compared to the single clause or phrase that typifies student justifications in the Geometry tutor. Consequently, we use an abductive theorem prover and a physics axiom set instead of a classifier and an ontology.

BEETLE [13] and PACO [22] are NL-based tutors for procedural tasks. The students need to do steps in a hierarchical, partial order which they may or may not know. A major challenge for these systems is allowing students to do steps when they can and offering hints when they can't. We have avoided (at least in this first version of the system) the problems of conducting a mixed initiative dialogue by giving the student all the initiative during the essay-entering phase, and having the tutor take the lead otherwise.

8 Conclusions

Our mission is to explore the cost-benefit space of NL-based tutoring. As an initial data point, we have constructed a baseline system and are evaluating it this spring. The evaluation compares Why2-Atlas to 3 other methods for teaching the same qualitative physics knowledge using the same 10 essay questions: (1) human tutors, (2) Why2-AutoTutor, a statistical NLP system described earlier, and (3) expository physics texts. Although Why2-Atlas is proving to be fast and robust enough, its knowledge sources are not nearly as well developed as we would like, so it often misunderstands the students. However, we anticipated this, and engineered its responses to yield reasonable dialogue despite its misunderstandings. We hypothesize that this baseline system should do about as well as Why2-AutoTutor. We expect both tutoring systems to be better than the expository texts, but not as good as the human tutors. Regardless of how this first evaluation turns out, we will use its log file data to improve our knowledge sources, then run the evaluation again. This will yield a second data point on the tradeoff between NLP sophistication and learning gains. Simultaneously, we are building tools that should lower the development effort for NL-based tutoring. Although the technology for NL-based tutoring is clearly still in its early stages, we hope to understand its benefits, its costs, and how to change both.

Acknowledgements. This research was supported by MURI grant N00014-00-1-0600 from ONR Cognitive Science, and by NSF grant 9720359. We're delighted to acknowledge the help of Art Graesser and his Tutoring Research Group, our partners in the Why2 project.

References

1. Graesser, A.C., et al., Using latent semantic analysis to evaluate the contributions of students in AutoTutor. *Interactive Learning Environments*, 2000.
2. Person, N.K., et al., Simulating human tutor dialog moves in AutoTutor. *International Journal of Artificial Intelligence in Education*, in press.
3. Hake, R.R., Interactive-engagement vs. traditional methods: A six-thousand student survey of mechanics test data for introductory physics students. *American Journal of Physics*, 1998. 66(4): p. 64-74.
4. Hestenes, D., M. Wells, and G. Swackhamer, Force concept inventory. *The Physics Teacher*, 1992. 30: p. 141-158.

5. Rose, C.P. A framework for robust semantic interpretation. In *The First Meeting of the North American Chapter of the Association for Computational Linguistics*. 2000.
6. Rose, C.P. A syntactic framework for semantic interpretation. In *The ESSLLI Workshop on Linguistic Theory and Grammar Implementation*. 2000.
7. Rose, C.P. and A. Lavie, Balancing robustness and efficiency in unification-augmented context-free parsers for large practical applications, in *Robustness in Language and Speech Technology*, J.C. Junqua and G.V. Noord, Editors. 2001, Kluwer Academic press.
8. McCallum, A.K., *Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering*. 1996, CMU: Pittsburgh, PA.
9. Halliday, M.A.K., *An Introduction to Functional Grammar*. 1985: Adward Arnold: A division of Hodder and Stoughton.
10. Bresnan, J., *The Mental Representation of Grammatical Relations*. 1982, Cambridge, MA: MIT Press.
11. Grishman, R., C. Mcleod, and A. Meyers. COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*. 1994.
12. Aleven, V., O. Popescu, and K.R. Koedinger. A tutorial dialogue system with knowledge-based understanding and classification of student explanations. In *Second IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*. 2001. Seattle, WA.
13. Core, M.G., J.D. Moore, and C. Zinn. Supporting constructive learning with a feedback planner. In *AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications*. 2000. Cape Cod, MA.
14. Jordan, P.W., et al. Engineering the Tacitus-lite weighted abduction inference engine for use in the Why-Atlas qualitative physics tutoring system. submitted.
15. Hobbs, J., et al., Interpretation as abduction. *Artificial Intelligence*, 1993. 63(1-2): p. 69-142.
16. Strube, M. Never look back: An alternative to centering. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the ACL*. 1998.
17. Jordan, P.W., C.P. Rose, and K. VanLehn, Tools for authoring tutorial dialogue knowledge, in *Artificial Intelligence in Education: AI-Ed in the Wired and Wireless future*, J.D. Moore, C. Redfield, and W.L. Johnson, Editors. 2001, IOS: Washington, DC. p. 222-233.
18. Graesser, A.C., et al., Intelligent tutoring systems with conversational dialogue. *AI Magazine*, 2001. 22(4).
19. Rose, C.P. Facilitating the rapid development of language understanding interfaces for tutoring systems. In *AAAI Fall Symposium on Building Tutorial Dialogue Systems*. 2000. Cape Cod, MA.
20. Evens, M.W., et al. Circsim-Tutor: An intelligent tutoring system using natural language dialogue. In *Twelfth Midwest AI and Cognitive Science Conference*. 2001. Oxford, OH.
21. Freedman, R., et al., ITS Tools for natural language dialogue: A domain-independent parser and planner, in *Intelligent Tutoring Systems: 5th International Conference, ITS 2000*, G. Gauthier, C. Frasson, and K. VanLehn, Editors. 2000, Springer: Berlin. p. 433-442.
22. Rickel, J., et al., Building a bridge between intelligent tutoring and collaborative dialogue systems, in *Artificial Intelligence in Education: AI-Ed in the Wired and Wireless Future*, J.D. Moore, C. Redfield, and W.L. Johnson, Editors. 2001, IOS: Washington, DC. p. 592-693.