Interpretation and Generation in a Knowledge-Based Tutorial System

Myroslava O. Dzikovska, Charles B. Callaway, Elaine Farrow
Human Communication Research Centre, University of Edinburgh
2 Buccleuch Place, Edinburgh, EH8 9LW,
United Kingdom,
{mdzikovs,ccallawa,efarrow}@inf.ed.ac.uk

Abstract

We discuss how deep interpretation and generation can be integrated with a knowledge representation designed for question answering to build a tutorial dialogue system. We use a knowledge representation known to perform well in answering exam-type questions and show that to support tutorial dialogue it needs additional features, in particular, compositional representations for interpretation and structured explanation representations.

1 Introduction

Human tutoring is known to help students learn compared with reading textbooks, producing up to two standard deviations in learning gain (Bloom, 1984). Tutorial systems, in particular cognitive tutors which model the inner state of a student's knowledge, help learning but result in only up to 1 standard deviation learning gain (Anderson et al., 1995). One current research hypothesis is that this difference is accounted for by interactive dialogue, which allows students to ask questions freely, and tutors to adapt their direct feedback and presentation style to the individual student's needs.

Adding natural language dialogue to a tutorial system is a complex task. Many existing tutorial dialogue systems rely on pre-authored curriculum scripts (Person et al., 2000) or finite-state machines (Rosé et al., 2001) without detailed knowledge representations. These systems are easy to design for curriculum providers, but offer limited flexibility because the writer has to predict all possible student questions and answers.

We argue that the ability to interpret novel, context-dependent student questions and answers, and offer tailored feedback and explanations is important in tutorial dialogue, and that a domain knowledge representation and reasoning engine is necessary to support these applications. We discuss our knowledge representation, and the issues of integrating it with state-of-the-art interpretation and generation components to build a knowledge-based tutorial dialogue system.

Our application domain is in basic electricity and electronics, specifically teaching a student how to predict the behavior and interpret measurements in series and parallel circuits. This is a conceptual domain - that is, students are primarily focused on learning concepts such as voltage and current, and their relationships with the real world. The students use a circuit simulator to build circuits, and their questions and answers depend on the current context.

There are various sources of context-dependency in our domain. Students and tutors refer to specific items in the simulation (*e.g.*, "Which lightbulbs will be lit in these circuits?"), and may phrase their answers in an unexpected way, for example, by saying "the lightbulbs in 2 and 4 will be out" instead of naming the lit lightbulbs. Moreover, students may build arbitrary circuits not included in the question, either because they make mistakes, or because a tutor instructs them to do so as part of remediation. Thus it would be difficult to produce and maintain a finite-state machine to predict all possible situations, both for interpreting the input and for generating feedback based on the state of the environment and the previous dialogue context: a domain reasoner is necessary to handle such unanticipated situations correctly.

We describe a tutorial system which uses a description logic-based knowledge representation to

generate intelligent explanations and answers to a student's questions, as well as to interpret the student's language at all stages of the dialogue. Our approach relies on using an existing wide-coverage parser for domain-independent syntactic parsing and semantic interpretation, as well as a wide-coverage deep generation system. We discuss the issues which arise in connecting such resources to a domain knowledge representation in a practical system.

2 Motivation

A good teaching method for basic electricity and electronics is eliciting cognitive dissonance (Schaffer and McDermott, 1992; Arnold and Millar, 1987) which we are implementing as a "predict-verify-evaluate" (PVE) cycle. The students are asked to make predictions about the behavior of a schematic circuit and then build it in a simulation environment. If the observed results do not match their predictions, a discussion ensues, where the computer tutor helps a student learn the relevant concepts. The PVE exercises are complemented with exercises asking the students to identify properties of circuits in diagrams and to interpret a circuit's behavior.

Thus, the system has to answer questions about circuits which students build and manipulate dynamically in a simulation environment, and produce explanations and feedback tailored to that individual context. This relies on the following system capabilities:

- *Understanding and giving explanations*. Since the system relies on inducing cognitive dissonance, it should be able to explain to the student why their prediction for a specific circuit was incorrect, and also verify explanations given by a student.
- *Unrestricted language input with reference resolution.* Similar to other conceptual domains (VanLehn et al., 2002) the language observed in corpus studies is varied and syntactically complex. Additionally, in our domain students refer to items on screen, e.g. "the lightbulb in 5", which requires the system to make the connection between the language descriptions and the actual objects in the environment.
- *Tailored generation*. The level of detail in the explanations offered should be sensitive to student knowledge of the domain. Tutorial utterances should be natural and use correct terminology even if a student doesn't.

To support answering questions and giving explanations, we chose the KM knowledge representation environment (Clark and Porter, 1999) as a basis for our implementation. KM is a description-logic based language which has been used to represent facts and rules in a HALO system for AP chemistry tests (Barker et al., 2004). It supports the generation of explanations and obtained the highest explanation scores in an independent evaluation based on an AP chemistry exam (Friedland et al., 2004). Thus it is a good choice to provide reasoning support for explanations and answering novel questions in a tutorial system. However, KM has not been used previously in connection with natural language input for question answering, and we discuss how the limitations of KM representations affect the interpretation process in Section 4.

We use a deep domain-independent parser and grammar to support language interpretation, and a deep generator to provide natural sounding and context-dependent text. Both deep parsing and generation provide the context adaptivity we need, but they are time-consuming to build for a specific domain. Now that a number of deep domain-independent parsing and generation systems are available in the community, our research goal is to investigate the issues in integrating them with the knowledge representation for question answering to support the requirements of a tutorial dialogue system. We focus on context-dependent explanation understanding and generation as a primary tutoring task in our domain. Section 3 discusses our representations, Section 4 presents the issues arising in knowledge representation to support interpretation, and Section 5 discusses the requirements for appropriate explanation generation and how it can be integrated into the system.

3 Representations

From the point of view of tutoring, the most important requirement on the knowledge representation is that system reasoning should closely match human reasoning, so that it can be explained to students in meaningful terms. Thus, for example, a numerical circuit simulator is well suited for dynamically displaying circuit behaviors, but not for conceptually tutoring basic circuits, because it

hides physics principles behind complex mathematical equations that are not suitable for learners.

To design our knowledge representation we started with a set of lessons for our domain designed by psychologists experienced in designing training courses for physics and simulated environments. The lessons were used in a data collection environment with experienced tutors conducting tutoring sessions over a text chat interface. Each student and tutor were required to go through the materials presented as a set of slides and solve pre-defined exercises, but the students asked questions to get help with problem-solving. The tutor had complete freedom to choose how to answer student questions and how to remediate when students made mistakes. We are using this data set to study the types of errors that students make as well as the language used by both students and tutors. The latter serves as a guide to developing our interpretation and generation components.

In addition to the set of materials, the course designers provided a "glossary" of concepts and facts that students need to learn and use in explanations, containing approximately 200 concepts and rules in a form which should be used in model explanations. We then developed our knowledge representation so that concepts listed in the glossary were represented as KM concepts, and facts are represented as rules for computing slots.

An example KM representation for our domain is shown in Figure 1. It represents the fact that a lightbulb will be on if it is in a complete path (i.e. a closed path containing a battery). The explanation is generated using the comment structure *[lightbulbstate]* shown in Figure 2 (slightly simplified for readability). Explanations are generated separately from reasoning in the KM system because reasoning in general contains too many low-level details. For example, our rule for computing a lightbulb state includes two facts: that the lightbulb has to be in a complete path with a battery, and that a lightbulb is always in one state only (i.e. it cannot be broken and on at the same time). The latter is required for proof completeness, but is too trivial to be mentioned to students. KM therefore requires knowledge engineers to explicitly designate the facts to be used in explanations.

This representation allows KM to generate detailed explanations by using a template string and then explaining the supporting facts. An example of a full explanation, together with the adjustments needed to use it in dialogue rather than as a complete answer, is given in Section 5.

Currently, KM only supports generating explanations, but not verifying them. The explanation mechanism produces explanations as text directly from the knowledge representation, as shown in Figure 2(a). This generation method is not well suited for a tutorial dialogue system, because it does not take context into account, as discussed in Section 5. Therefore, we are designing a structured representation for explanations to be produced by the KM explanation mechanism instead of using English sentences, shown in Figure 2(b). This will allow us to generate more flexible explanations (Section 5) and also to interpret student explanations (Section 4).

4 Interpretation

The interpretation process consists of parsing, reference resolution, dialogue act recognition and diagnosing student answers. We discuss reference resolution and diagnosis here as these are the two steps impacted by the knowledge representation issues. As a basic example we will use the student answer from the following pair:[1]

Problem: For each circuit, which lightbulbs will be lit? Explain.

Student: the bulbs in 1 and 3 are lit because they are in a closed path with a battery

To respond to this answer properly, the system must complete at least the following tasks. First, it must resolve "the bulbs in 1 and 3" to corresponding object IDs in the knowledge base, for example, *LB-13-1-1* and *LB-13-3-1*. Then, it must verify that the student statement is factually correct. This includes verifying that the lightbulbs in 1 and 3 will be lit, and that each of them is in a closed path with a battery. Finally, it must verify that the student explanation is correct. This is separate from verifying factual correctness. For example, a statement "because they are in a closed path" is true for both of those lightbulbs, but it is not a complete explanation, because a lightbulb may be in a closed path which does not contain a battery, where it won't be lit.

[1] These utterances come from our corpus, though most of the student answers are not as easy to parse. We are working on robust parsing methods to address the issues in parsing less coherent utterances.

```
(every LightBulb has
  (state ((must-be-a Electrical-Usage-State) (exactly 1 Electrical-Usage-State)
        (if (the is-damaged of Self) then *Broken-Usage-State
          else (if (has-value (oneof ?batt in (the powered-by of Self)))
                                  where ((the state of ?batt) = *Charged-Power-State)))
               then *On-Usage-State else *Off-Usage-State) [lightbulbstate])))
```

Figure 1: The representation of a lightbulb in our KM database

```
(a)(comment [lightbulbstate]
    (:sentence ("a working lightbulb is on if it is in a complete path with a charged battery"))
    (:supporting-facts (:triple Self powered-by *) (forall (the powered-by of Self) (:triple It state *)))
(b)(comment [lightbulbstate]
    (:rule :object LightBulb :fact (?lb state *On-Usage-State)
      :requires ((?lb powered-by ?v1) (?v1 instance-of Battery) (?v1 state *Charged-Power-State))
      :bindings ((?lb ⟨ Self ⟩) (?v1 ⟨ the powered-by of Self ⟩)) )
```

Figure 2: A sample comment structure to generate an explanation for a lit lightbulb (a) The KM text template (b) a new structured representation. Items in angle brackets are computed dynamically

4.1 Interpreting Factual Statements

We use the TRIPS dialogue parser (Dzikovska, 2004) for interpretation. The TRIPS parser provides a two-layer architecture where the utterance meaning is represented using a domain-independent semantic ontology and syntax. The domain-independent representation is used for discourse processing tasks such as reference resolution, but it is connected to the domain-specific knowledge representation by mapping between the domain-independent and domain-specific ontologies (Dzikovska et al., 2003; Dzikovska, 2004). This architecture allows us to separate linguistic and domain-specific knowledge and easily specialize to new domains.

When applied in our domain, the TRIPS interpretation architecture was helpful in getting the interpretation started quickly, because we only needed to extend the lexicon with specific terms related to basic electricity and electronics (*e.g.*, "multimeter"), while other lexical items and syntactic constructions were provided in the domain-independent part.

The reference resolution module operates on TRIPS domain-independent representations sending queries to KM as necessary, because the TRIPS representations offer linguistic features to guide reference resolution not available in the representations used for reasoning. We use a recursive reference resolution algorithm similar to Byron (2002) which first resolves "1 and 3" are resolved as names for *Circuit-13-1* and *Circuit-13-3*,[2] and then queries KM to find all lightbulbs in those circuits. Dialogue context is used to interpret the reference resolution results. In this case, the context does not matter because the question sets up all lightbulbs on screen as contextually relevant. But if the student had said "the ones in 1 and 3", the query would be for all components in circuits 1 and 3, and then our algorithm will filter the query results based on the question context to retain only lightbulbs.

Once the references are resolved, the whole sentence is converted to a KM statement which represents the student utterance, in our case *(the state of LB13-1-1) = *On-Usage-State*, where *LB13-1-1* is the lightbulb obtained by reference resolution. This statement is sent to the KM system, which verifies that it is correct. This procedure allows us to use dialogue context in understanding, and also to check correctness of answers easily, even if they are phrased in an unanticipated way.

However, even with the layer of separation of linguistic and domain knowledge provided by the TRIPS architecture, we found that the need to support interpretation in a compositional way influences the interaction with knowledge representation. There are many ways to express the same query to KM, which differ in efficiency. Two ex-

[2] This step is not trivial, because on other slides the label "1" refers to terminals or other components rather than whole circuits, and therefore there is no 1-to-1 correspondence between names and objects in the environment.

(a) (allof ?x in (the all-instances of LightBulb) where ((the components of Circuit-13-1) include ?x))
(allof ?x (LightBulb ?x) and (components Circuit-13-1 ?x))
(b) (allof ?comp in (the components of Circuit-13-1) where (?comp isa LightBulb))
(allof ?x (components Circuit-13-1 ?x) and (LightBulb ?x))

Figure 3: KM Queries to to retrieve all lightbulbs in a circuit with corresponding first-order logic glosses.

ample queries to ask the same question are given in Figure 3. While their first order logic semantics is equivalent except for the order of conjuncts, they are expressed in a very different way in the KM syntax. Version (b) is more efficient to ask, because it retrieves the components of circuit 1 first, a smaller set than the set of all lightbulbs.

This asymmetry presents a challenge to both language interpretation and knowledge engineering. Existing reference resolution algorithms (Byron, 2002; Bos, 2004) expect the queries for "the lightbulb" and "the lightbulb in 1" to be strictly compositional in the sense that the phrase "the lightbulb" will be represented identically in both cases, and "in 1" is represented as an additional constraint on the lightbulbs. This corresponds to the query variant (a) in the system. Otherwise a large amount of query-specific transformations may be required to produce queries for complex noun phrase descriptions, diminishing the scalability of the approach.

We had to spend a significant portion of time in the project developing an efficient and compositional knowledge representation. Our current solution is to prefer compositionality over efficiency, even though it impacts performance in some cases, but we are working on a more general solution. Instead of converting directly to KM from domain-independent language representations, we will convert all queries in a FOL-like syntax shown in Figure 3 which uses concepts from the KM representation, but where all conjuncts are treated identically in the syntax. The problem of converting this representation to the optimal KM form can then be seen as an instance of query optimization. For example, we can reorder the conjuncts putting the relations which include an instance constant (*e.g.*, (the components of Circuit-13-1)) first in the query, because they are more likely to limit the search to small sets of objects. This representation can be easily converted in the KM syntax, and is also useful for explanation understanding and generation as discussed below.

4.2 Explanation Understanding

While KM has facilities for generating explanations, it does not have support for reading in a student explanation and verifying it. We devised a method to support this functionality with the aid of KM explanation generation mechanism. Any time a student offers an explanation, the KM reasoner will be called to generate its own explanation for the same fact, in the structured format shown in Figure 2(b). Then the student explanation (converted into the same intermediate syntax) can be matched against the KM-generated explanation to verify that it is complete, or else that certain parts are missing.

In our example, the student explanation "because they are in a closed path with a battery" will be represented as *(?pa instance-of Path) (?pa is-closed t) (?b instance-of Battery) (?pa contains ?b) (?pa contains LB-13-1-1)*.[3] This explanation does not directly match into the explanation structure from Figure 2(b), because it uses the more specific term "in closed path with a battery" rather than the more general term "in complete path" (represented by the *powered-by* slot). However, as part of generating the explanation, an explanation structure for the *powered-by* will be generated, and it will include the facts *(?pa is-closed t) (?pa contains ?b)*. This will match the student explanation. It will be up to the tutorial module to decide whether to accept the explanation "as is", or lead the student to use the more precise terminology, as discussed in Section 5.

This method can address student explanations as long as they correspond to parts of typical explanations, and identify missing parts. The biggest open problem we face is equivalent inferences. For example, a student may say "A lightbulb is not on" instead of "a lightbulb is off". KM reasoning handles those differences when verifying factual correctness, but KM does not support similar reasoning for matching explanations (which

[3] Here "they" would be resolved first to a set of lightbulbs, and each instance will be treated separately to verify that the explanation applies.

would correspond to verifying full proofs rather than individual facts). We are considering bringing a theorem prover to reason over intermediate representations together with KM axioms to help interpret explanations, as done in (Makatchev et al., 2004; Bos, 2005).

5 Generation

The task of the utterance generation component is to produce tutorial dialogue, such as asking new questions of the student, conveying the correctness of their answers, and giving explanations. Explanations may be given in response to a student's direct 'why' question or when a student has erred and the pedagogical reasoner has decided that an explanation is the best remediation strategy. In each case, the utterance generator must not only provide a correct, thorough and coherent explanation, but must tailor it so that the student doesn't receive too much or too little information. To be tailorable, explanations must be derived from the represented domain knowledge and from what the tutoring system knows about the student (e.g., their recent performance).

Directly producing explanations by appending together pieces of hand-written strings as in Figure 2(a) usually results in long explanations that contain little detail of interest to the student. Figure 4 contains one such example explanation generated by the KM system in our domain and derived from a query based on the production rule in Figure 1. This explanation makes sense in answering an exam question, as intended in the KM system, but it is not necessarily helpful in dialogue.

As an example, suppose the student had incorrectly answered the question in Section 4, and the tutoring system decides to correctly explain why the lightbulbs are lit. Usually, a full explanation is not necessary in these cases. In the case where a student gave an incomplete explanation, namely leaving out the necessary mention of the battery, a simple response of the form "Yes, but don't forget the battery" will be infinitely more helpful than the full explanation. If the student's explanation is completely correct, but they have failed to notice a change in the environment, the more appropriate explanation is "The lightbulb is in a closed path, as well as the battery, but the battery is not operational". Furthermore, if a student has shown that they are knowledgeable about certain fundamental facts, such as what states a lightbulb may be in, statements like "A lightbulb can be on, off or broken" should be removed.

Adding this reasoning directly to the knowledge base would make it unwieldy and unmodifiable, and the string-based generation in KM comments does not allow for adapting explanations based on external knowledge such as a student model. To adapt the KM explanation mechanism to support such context-dependent generation, instead of creating explanations via template strings, we have devised the representation presented in Figure 2(b) that is based on semantics and allows us to modify an explanation after it has been produced by the KM reasoning process but before it has been converted into a string representation.

Based on this semantic representation, explanation content can be selected more appropriately. If the interpreter discussed in Section 4.2 determines that parts of the explanation from the *:requires* field are missing, the generation can focus only on that part of the explanation. The requirements list would also be used to determine if the student is not aware of environment properties, such as that a battery is damaged. Finally, the facts known to the student can be removed if the corresponding semantic forms were used in previous explanations.

In addition to selecting the explanation content properly, it is important that the responses given to the student sound fluid and are easy to understand. In dialogue, in particular, it is important that pronouns can be generated based on references important for the student, and avoid repetitiveness in syntax. Knowledge of linguistic features such as number and gender, and also knowledge of what was previously mentioned in the discourse, is necessary to support such natural text generation.

Deep generation utilizes this represented knowledge along with grammatical and lexical knowledge of a language, rather than hand-written strings, to produce utterances. Our current implementation uses a custom utterance generation component and the STORYBOOK (Callaway and Lester, 2002) deep text generator modified to work in a dialogue context. Once the explanation content is selected, it is passed to the STORYBOOK system to produce the actual utterance text.

6 Discussion and Related Work

Existing tutorial dialogue systems most often rely on one of two approaches for interpretation: they either use wide coverage but shallow language

```
A lightbulb can be on, off or broken.
A working lightbulb is on if it is in a complete path with a charged battery.
  The complete paths of a component are those which are valid, closed, and complete.
    A path is complete if it is a closed path with at least one battery and at least...
      A path is closed if it is a valid path, a sequence of more than two terminals, ...
        A path is valid if it is a single sequence with more than one terminal, all ...
        The path (:seq t1-13-1-3 t1-13-1-2 t1-13-1-1 t1-13-1-4) is valid.
      The path (:seq t1-13-1-3 t1-13-1-2 t1-13-1-1 t1-13-1-4) is closed.
        ... 6 lines showing that the path contains both L1-13-1-1 and B1-13-1-1 ...
    The path (:seq t1-13-1-3 t1-13-1-2 t1-13-1-1 t1-13-1-4) is complete.
  L1-13-1-1 is in a complete path with B1-13-1-1.
  A battery is charged unless it is damaged.
  B1-13-1-1 is charged.
L1-13-1-1 is on.
```

Figure 4: Untailored text produced by appending strings from production rules.

interpretation techniques (e.g. LSA (Person et al., 2000), finite-state parsing (Glass, 2001)) in combination with shallow knowledge representations (tutorial scripts or FSA-based knowledge construction dialogues), or they use deep KR&R systems but with highly domain-specific parsing and semantic interpretation (e.g. ATLAS-ANDES (Rosé et al., 2001), PACT (Aleven et al., 2002)).

The Why2-Atlas system (VanLehn et al., 2002) makes progress on combining wide coverage interpretation with deep knowledge representation by utilizing a wide-coverage syntactic grammar (Rosé, 2000) and a theorem prover to interpret student essays (Makatchev et al., 2004). However, once the misconceptions are diagnosed, the remediation is done via KCDs, with very limited language input and pre-authored responses, and without allowing students to ask questions. Our approach attempts to address issues which arise in making remediation more flexible and dependent on context, while still relying on wide-coverage language interpretation and generation.

The issues we encountered in integrating compositional interpretation and reference resolution with efficient knowledge representation is similar to a known problem in natural language interfaces to databases which may contain slots with complex meanings. (Stallard, 1986) solves this problem by providing inference schemas linking complex-valued slots with compositional representations. Our solution in mapping domain-independent to domain-specific representation is similar, but stricter compositionality is needed for reference resolution support, placing additional constraints on knowledge engineering as we discussed in Section 4.

We glossed over the interpretation issues related to metonymy and other imprecise formulations in questions (Aleven et al., 2002). A taxonomy of imprecise manual question encodings by domain experts is presented in (Fan and Porter, 2004). They also propose an algorithm to address loosely encoded questions using ontological knowledge. This algorithm in effect performs question interpretation, and we are planning to incorporate it into our interpretation mechanism to help interpret question representations obtained automatically during language interpretation.

Text generation of the type that can handle the necessary linguistic phenomena needed have not been implemented in tutoring systems that use dialogue. The DIAG-NLP tutorial dialogue system (Eugenio et al., 2005) shows that structured explanations from deep generation supported by knowledge representation and reasoning improve learning. However, it does not engage in a dialogue with the user, and in this paper we showed that explanations need to be further adjusted in dialogue based on previous student responses and knowledge. Deep generation using context has been used in some other types of dialogue systems such as collaborative problem solving (Stent, 2001), and we expect that the approaches used in content selection and planning in those systems will also transfer to our deep generation system.

7 Conclusions

We discussed the implementation of a tutorial dialogue system which relies on a domain knowledge representation to verify student answers and offer appropriate explanations. Integration with domain-independent interpretation and generation components places additional requirements on knowledge representation, and we showed how an existing knowledge representation mechanisms used in answering exam questions can be adapted to the more complex task of tutoring, including interpreting student explanations and generating ap-

propriate feedback.

Acknowledgments

This material is based upon work supported by a grant from The Office of Naval Research number N000149910165.

References

V. Aleven, O. Popescu, and K. Koedinger. 2002. Pilot-testing a tutorial dialogue system that supports self-explanation. *Lecture Notes in Computer Science*, 2363.

J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. 1995. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207.

M. Arnold and R. Millar. 1987. Being constructive: An alternative approach to the teaching of introductory ideas in electricity. *International Journal of Science Education*, 9:553–563.

K. Barker, V. K. Chaudhri, S. Y. Chaw, P. Clark, J. Fan, D. Israel, S. Mishra, B. W. Porter, P. Romero, D. Tecuci, and P. Z. Yeh. 2004. A question-answering system for AP chemistry: Assessing KR&R technologies. In *KR*, pages 488–497.

B. S. Bloom. 1984. The two sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13:3–16.

Johan Bos. 2004. Computational semantics in discourse: Underspecification, resolution, and inference. *Journal of Logic, Language and Information*, 13(2):139–157.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics (IWCS-6)*.

Donna K. Byron. 2002. *Resolving Pronominal Reference to Abstract Entities*. Ph.D. thesis, University of Rochester.

Charles B. Callaway and James C. Lester. 2002. Narrative prose generation. *Artificial Intelligence*, 139(2):213–252, August.

P. Clark and B. Porter, 1999. *KM (1.4): Users Manual*. http://www.cs.utexas.edu/users/mfkb/km.

M. O. Dzikovska, M. D. Swift, and J. F. Allen. 2003. Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains. In *Proceedings of IJCAI-03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

M. O. Dzikovska. 2004. *A Practical Semantic Representation For Natural Language Parsing*. Ph.D. thesis, University of Rochester.

B. Di Eugenio, D. Fossati, D. Yu, S. Haller, and M. Glass. 2005. Natural language generation for intelligent tutoring systems: A case study. In *12th International Conference on Artificial Intelligence in Education*, pages 217–224.

J. Fan and B. W. Porter. 2004. Interpreting loosely encoded questions. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence*, pages 399–405.

N. S. Friedland, P. G. Allen, M. J. Witbrock, G. Matthews, N. Salay, P. Miraglia, J. Angele, S. Staab, D. Israel, V. K. Chaudhri, B. W. Porter, K. Barker, and P. Clark. 2004. Towards a quantitative, platform-independent analysis of knowledge systems. In *KR*, pages 507–515.

M. Glass. 2001. Processing language input in the CIRCSIM-tutor intelligent tutoring system. In J. Moore, C. L. Redfield, and W. L. Johnson, editors, *Artificial Intelligence in Education*. IOS press.

M. Makatchev, P. W. Jordan, and K. VanLehn. 2004. Abductive theorem proving for analyzing student explanations to guide feedback in intelligent tutoring systems. *J. Autom. Reasoning*, 32(3):187–226.

N. Person, A.C. Graesser, D. Harter, and E. Mathews. 2000. Dialog move generation and conversation management in autotutor. In *Workshop Notes of the AAAI '00 Fall Symposium on Building Dialogue Systems for Tutorial Applications*.

C. Rosé, P. Jordan, M. Ringenberg, S. Siler, K. VanLehn, and A. Weinstein. 2001. Interactive conceptual tutoring in atlas-andes. In *Proceedings of AI in Education 2001 Conference*.

C. Rosé. 2000. A framework for robust semantic interpretation. In *Proceedings 1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

P. S. Schaffer and L. C. McDermott. 1992. Research as a guide for curriculum development: An example from introductory electricity. part ii: Design of instructional strategies. *American Journal of Physics*, 60(11):1003–1013.

D. G. Stallard. 1986. A terminological simplification transformation for natural language question-answering systems. In *ACL Proceedings, 24th Annual Meeting*, pages 241–246.

A. Stent. 2001. *Dialogue Systems as Conversational Partners: Applying conversation acts theory to natural language generation for task-oriented mixed-initiative spoken dialogue*. Ph.D. thesis, University of Rochester, Rochester, NY, August.

K. VanLehn, P. Jordan, C. P. Rosé, and The Natural Language Tutoring Group. 2002. The architecture of why2-atlas: a coach for qualitative physics essay writing. In *Proceedings of Intelligent Tutoring Systems Conference*.