AIED 05 WORKSHOP 8

Amsterdam, 18-22 July, 2005

Mixed Language Explanations in Learning Environments

AIED 05 Amsterdam WORKSHOP

12th International Conference on Artificial Intelligence in Education, Amsterdam, the Netherlands

Workshop scientific committee

Program chairs:

Committee members:

Contents

Foreword

The focus of this workshop is on the issues of and methods for analyzing and generating mixed language explanations, namely, explanations that combine natural language, symbolic (e.g., algebraic) expressions, formal domain concepts, and diagrams, in learning environments.

There are many aspects of human-human learning interactions that can contribute to an assessment of a student's knowledge and ability level. The equations and diagrams the student creates while solving a problem are just one type of interaction language that can influence an evaluation of the student. Questions and comments that the student produces during the problem solving while deciding what equations and diagrams to generate, and the student's responses to questions and guidance, can be revealing as well. In learning systems, however, student assessment has been primarily limited to interpreting equation and diagram manipulations, and the timing and type of help requests a student makes relative to a menu of available help.

Many efforts are under way to increase the ways in which students can communicate with a learning environment. One area of exploration is the addition of both spoken and written natural language understanding and generation capabilities to equation and diagram manipulation interfaces. Mixing natural language with algebraic and graphical forms of communication demonstrates that each cannot be analyzed completely separately and then combined, since each can influence the analysis of the others. For example, an explicit or implicit presence of equations, diagrams, and formal domain concepts can heavily influence the content and form of the natural language used by the student and the system, and vice versa, so that the natural language used may

- refer to algebraic, conceptual, and diagrammatic domain entities,
- describe or explain algebraic, conceptual, and diagrammatic domain entities,
- refer to or describe past/future actions or cognitive states (e.g., "I don't understand this equation."),
- collapse reasoning steps that previously would have been expanded, in the absence of NL, to demonstrate the student's ability and understanding.

Most of these phenomena have been observed in systems that

- combine algebraic input methods with natural language textual explanations in interactive proof environments,
- combine a diagrammatic interface with natural language,
- and even systems that deploy only natural language interfaces, but in a context that requires building mental models of algebraic expressions, formal concepts, or diagrams.

The analysis of student input in these mixed language systems generally aims at (1) verifying whether the student's input is correct and (2) diagnosing possible sources of errors if it is incorrect. There is still the usual difficulty of verification and diagnosis being NP-complete tasks that must rely on heuristics and additional constraints to be feasible for implementation. But now there is the added problem of needing to reason about entities that can be realized in multiple ways, i.e., with algebraic expressions, diagrams, or natural language.

Unmodified, domain independent statistical methods that are typically used for analyzing natural language in current learning environments and that are the focus of much of today's applied research in computational linguistics for both analysis and generation do not provide the accuracy necessary for analyzing and generating descriptions of and references to precise objects like diagrams, algebraic expressions, and formally defined domain concepts. Formal methods for analyzing algebraic expressions, formal concepts, and diagrams, on the other hand, rely on input that is assumed to be 100% accurately recognized and this is not normally achievable with the uncertainty inherent in NLP.

The eleven papers that will be the focus of discussion in this workshop cover a wide range of domains and mixed language phenomena and describe both works in progress and deployed applications. The themes of the papers include

- issues involved in analyzing mixed language explanations,
- techniques for analyzing mixed language explanations,
- descriptions of learning environments and applications in which mixed-language use arises,
- descriptions of learning environments in which permitting mixed-language usage could be beneficial.

We thank the members of the program committee who worked under tight deadlines to provide each paper with multiple, thoughtful reviews. We anticipate that the reviews will also help stimulate discussions during the workshop. Further, we thank the authors for their contributions to this workshop. Finally, we thank the AIED conference organizers and workshops chairs for their support.

Pamela W. Jordan, Maxim Makatchev, Peter Wiemer-Hastings
May 25, 2005

Exploring the Instructional Effectiveness of Mixed-Language Peer Problem Solving Interactions

Gahgene GWEON, Carolyn ROSÉ, Regan CAREY, Zachary ZAISS
Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213

Abstract. This paper presents a controlled investigation of collaborative problem solving in which the behavior of a confederate peer learner is manipulated in order to measure the impact of the confederate peer learner's behavior on the behavior and learning of the student participant. The independent variables in the 2X2 factorial design include level of engagement (Lazy versus Engaged) and accuracy of problem solving contributions (High performing versus Low performing). We did not find evidence that the errors contributed by the confederate peer learners were harmful to student subjects working with them except in the case of students paired with Engaged Low performing peer learners. On the contrary, we found a small but reliable interaction effect in which students paired with Lazy Low performing peer learners derived some benefit from the errors they were exposed to whereas students paired with Engaged Low performing peer learners were harmed by the errors they were exposed to.

Introduction

Achieving the elusive "2 sigma effect" with intelligent tutoring technology has long been the holy grail of the field of intelligent tutoring research (Bloom, 1984). The search for the answer to this mystery has taken many forms, but one of the re-occuring trends through generations of investigation has been the somewhat naive belief that the answer lies in "humanizing" the technology, or creating analogs of human-human interaction contexts in which high levels of student learning have been observed. Two such paradigms are that of tutorial dialogue systems, in which the technology is modeled after a human tutor, and that of learning companions, where the technology is modeled after a student.

The rationale for learning companion technology grows out of the successful track record for the collaborative learning paradigm (Sharan, 1980). However, what is known about the mechanisms responsible for its success are largely at the group level rather than at the individual level. Even studies presenting evidence about specific effective patterns of interaction have largely provided correlational evidence, and thus do not offer insights on the causal mechanisms at work on the level of the individual student. Thus, what is known is not sufficient for guiding the design of artificial agents that can interact with students in way that yields the same effect. Consistent with recent arguments put forward regarding the debate between the situated versus cognitive perspective on learning research (Anderson, Reder, & Simon, 1997), we argue that what is needed as a next step is a careful investigation using controlled experimentation to construct a causal model of how specific features of an agent's behavior influence an individual student's behavior and learning. We outline our current research agenda and present results from a recent study illustrating our proposed methodology.

1. A Historical Perspective

The research literature investigating the construction of tutorial dialogue and learning companion environments present parallel experiences in attempting to emulate in technology what has been observed to be effective for learning in human-human scenarios.

Research towards developing tutorial dialogue systems has a longer history than learning companion research, extending back to the late 60s. While early efforts to emulate the effectiveness of human tutorial dialogue, such as the SCHOLAR system (Carbonell, 1969), the original WHY system (Stevens and Collins, 1977), and SOPHIE (Brown, Burton, & deKleer, 1982), were landmark systems in the history of intelligent tutoring research, they were naïve both theoretically and technologically. With the dawn of the 21st century, when both the fields of intelligent tutoring and language technologies had substantially matured, there was a resurgence of interest in bringing these two communities together once again. Wielding state-of-the-art language technology, intelligent tutoring researchers have made great strides in building and evaluating tutorial dialogue systems with students, often in realistic educational settings (Graesser, Bowers, Hacker, & Person, 1998; Rosé et al., 2001; Graesser, VanLehn, the TRG, & the NLT, 2002; Aleven, Koedinger, & Popescu, 2003; Evens and Michael, 2003). However, these systems, while offering strong proof of the technical feasibility of supporting meaningful interactions in natural language, have not yet yielded the dramatic improvements over more standard forms of tutoring systems that were predicted.

Just as tutorial dialogue system research grows out of the literature on successful human tutoring, work on learning companion technology grows out of the collaborative learning literature. When students work with tutoring systems that offer scaffolding and help on demand, the scaffolding creates a situation in which the student and the system are working together in some sense to solve a problem, although the collaboration is not overt or anthropomorphized. The idea behind learning companion technology has been to make that "collaboration" between the student and the machine overt by anthropomorphizing the help system in order to create a computer analogue of human-human collaborative learning. Students are meant to solve problems with intelligent tutoring systems as independently as possible, with the hints and scaffolding used only as needed. However, these passive and readily available resources can easily be abused (Baker et al., 2004). Students can cheat themselves out of learning from their problem solving experience by "gaming" the system's help functions at the places in the problem solving process that are most valuable for their learning (Baker et al., 2004). One hope of learning companion technology was to counter this tendency by holding students accountable, and thus foster engagement and deeper learning with technology. Some recent peer collaborative agents have been used successfully to monitor student behavior in on-line chat environments to detect when it becomes unproductive and explicitly offer a productive alternative (Vizcaino & du Boulay, 2002).

Previous approaches to adapting intelligent tutoring systems to simulate a collaborative learning setting include three paradigms: "computer as a co-learner" (Dillenbourg & Self, 1992), "learning companion systems" (Chan & Baskin, 1988) and "learning by teaching" (Chan & Baskin, 1988; Palthepu et al., 1991). Betty's Brain (Leelawong et al., 2002; Davis et al., 2003) is a teachable agent designed to engage students in learning oriented activities such as requesting explanations and building qualitative reasoning representations. However, its learning oriented functions are passive in a similar way to traditional help systems and other collaborative agents such as (Hietala & Niemirepo, 1998), and while use of them correlates with positive learning and performance outcomes, students have been observed to make use of them in a performance oriented fashion. Just as a shift towards interaction in natural language has not been enough to yield “the 2 sigma effect” with tutorial dialogue systems, simply anthropomorphizing help systems has not been sufficient for radically reshaping student behavior with intelligent tutoring systems.

2. Experimental Methodology

A key aspect of our research agenda is to investigate previous claims about best practices in learning companion design that have not been subjected to rigorous evaluation. We do this using a particular experimental design methodology, which provides a highly controlled way to examine mechanisms by which one peer learner’s behavior influences a partner learner’s behavior and learning. Specifically, we make use of confederate peer learners who are experimenters acting as peer learners but behaving in a highly scripted way. Building on the work of Hietala & Niemirepo (1998) contrasting high and low performing peer agents, we address the following questions in the study reported in this paper: (1) Under what circumstances do the errors that arise during collaborative problem solving interactions have a harmful (or helpful) effect on student learning? (2) How does the accuracy and the level of initiative taking of a peer learner’s contributions affect initiative taking in their partner?

Experimental Procedure: The experimental procedure consisted of 5 phases, consisting of three test phases alternating with two instructional phases. The experimental manipulation took place during phase 4. During the pre-instructional testing phase (phase 1), students filled out a consent form, took a pretest to assess their prior domain specific knowledge (for 15 minutes), and read the instructions for the first instructional phase. During the first instructional phase (phase 2), which was a human tutoring phase lasting 45 minutes, students received tutoring on the general concept of differentiation as well as 7 specific rules of differentiation from a human tutor. The tutor was blind to the student’s condition and adhered to a rigid schedule for covering all of the content in a consistent way between students. During the mid-instructional testing phase (phase 3), students took a short middle test to assess their learning during phase 2 (for 10 minutes). They also read the instructions for the second instructional phase. The second instructional phase (phase 4), was a problem solving phase where students worked through as many of 12 multi-step derivation problems as possible during the allotted 35 minutes. Finally, in the post-instructional phase (phase 5), students took the post-test (for 15 minutes) and filled out a questionnaire.

Materials: The materials for the experiments consisted of the following: (1) An 8 page web based lesson on derivatives provided material for the tutor and student to work through during Phase 2. It consisted of an overview and individual units on each of 7 specific rules of derivation. (2) 12 on-line problem solving exercises for Phase 4 requiring multiple rule applications. (3) 2 extensive tests (Test A and Test B) were used for the pre-test (in Phase 1) and the post-test (in Phase 5). These tests each consisted of 7 algebraic manipulation problems, 7 simple calculus problems to test knowledge of each specific differentiation rule, and 6 complex calculus problems requiring both multiple rule applications and algebra. We counterbalanced the order of the tests. In Phase 3, students took a middle test with 8 simple calculus problems, analogous to the second section of tests A and B, and three complex calculus problems requiring multiple rule applications.

Experimental Setup: All on-line problem solving was done using a structured problem solving interface designed for solving differentiation problems. Students first select a rule from a menu. Based on their selection, some explanation about the rule and slots to fill in were presented to the student. In some cases, additional menus were presented, allowing for embedded rule applications. No feedback was provided by the system based on the students' selections from the menu or entries in the text input boxes during the problem solving process. When the student or pairs of students were satisfied with their solution, they submitted it. If it was incorrect, they were then shown their incorrect derivation next to the correct one as a worked example including both the derivation and some explanation. The purpose was for them to compare and see how the problem should have been worked out and where their mistake occurred. In the case of a correct submission, the students moved on to the next problem. Students were located in different rooms. They could view and manipulate the same problem solving interface using VNC. Students communicated with each other by means of an MSN like typed chat interface. Their discussion included both text and equations. In the dialogue they discussed their division of labor and strategies for solving the problems.

Experimental manipulation: The experimental manipulation consisted of 4 conditions resulting from a 2X2 full factorial design with two factors describing characteristics of a scripted confederate peer problem solver, namely Lazy(LA)/Engaged(EN), referring to the frequency of the confederate problem solver's contributions to the problem solving process and High(HI)/Low(LO) referring to the accuracy of the confederate peer learner's contributions. During this phase of the experiment, one member of our team acted as a confederate student and another kept track of score, timing, and distribution of labor. The confederate student acted according to the following rules:

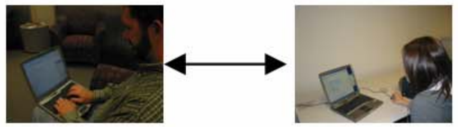
- LA/EN: In the Lazy condition, the confederate student contributed to solving the problem either by offering part of the solution in the chat window or by performing an action in the problem solving interface every 45 seconds. In the Engaged, condition, the confederate peer learner contributed every 8 seconds.
- HI/LO: In the High performing condition, the confederate student provided only correct contributions. In the Low performing condition, the confederate student provided incorrect contributions 2/3 of the time.

Subjects: 36 Carnegie Mellon students and staff participated in the study, randomly assigned to conditions: 58% male and 42% female, equally distributed between conditions.

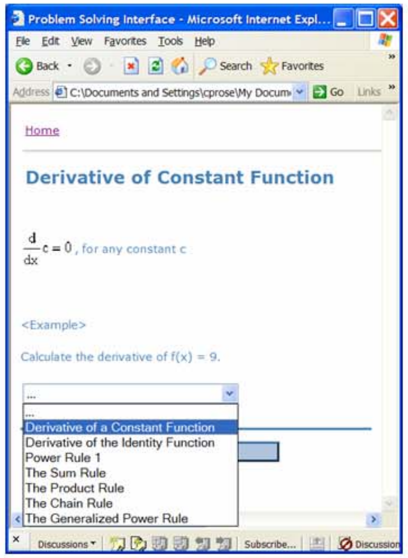
Results

Since the purpose of the 2X2 factorial experimental manipulation was to contrast the instructional impact of working with peer problem solvers with specific characteristics, we first verified the benefit of working with a peer problem solver over working alone. Using an ANCOVA with Post-test score as the dependent variable, Condition (SOL versus P2P) as the independent variable, and Pre-test score as a covariate, we verified that students in the P2P condition learned more than their peers in the SOL condition $F(1,18) = 6.0$, $p<.05$, $MSE = 5.64$, effect size = 1.1 standard deviations. We did not use the mid-test score as a covariate along with pretest score because it was not reliably correlated with post-test score with this population of students when we first factored out the effect of pretest score.

In Phase 1, students are told they will learn with a tutor in preparation for a problem solving contest



Tutor: the first rule we will study is called "constant rule". what do you think this rule says?
Student: i'm not entirely sure
Tutor: are you familiar with the symbol d/dx? or f ' (x)?
Student: yes, it's just been a while :)
Tutor: =) d/dx and f '(x) both mean "derivative of"
Student: ok
Tutor: so the first line says d(c)/dx=0. can you guess what that would mean?
Student: the slope for the constant (which is a function) is always 0?


Problem Solving Interface - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\cprose\My Docum Go Links
Home
Discussions Subscribe... Discussion

Derivative of Constant Function

$\frac{d}{dx}c = 0$, for any constant c

<Example>

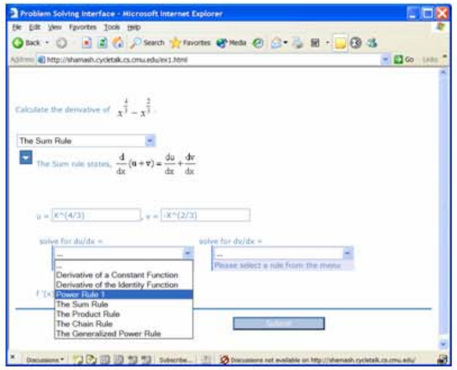
Calculate the derivative of f(x) = 9.

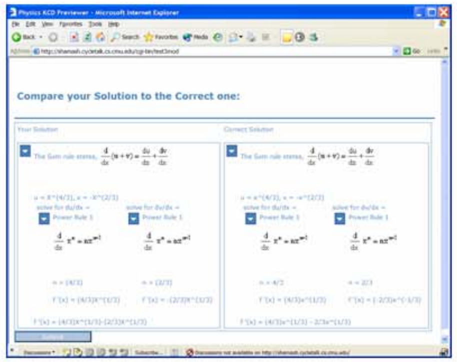
...
Derivative of a Constant Function
Derivative of the Identity Function
Power Rule 1
The Sum Rule
The Product Rule
The Chain Rule
The Generalized Power Rule

In Phase 2, students work together on a series of problems

PeerLearner: okay.. i think that's the answer, what do you think?
RealStudent: I have no idea
PeerLearner: **so let's try submitting then**
...
PeerLearner: **damn it...we got it wrong again... i think we almost had it, right? we got product rule**
RealStudent: shall we move on?
PeerLearner: but do you understand this?
RealStudent: **hell no**
PeerLearner: cause we're going to keep getting things wrong if we don't understand... so should we study this a little maybe?
RealStudent: sounds like a plan :)

PeerLearner: i think we got du/dx part right.. but we got dv/dx wrong.


Problem Solving Interface - Microsoft Internet Explorer
Calculate the derivative of
The Sum Rule
The Sum rule states,
solve for du/dx =
solve for dv/dx =
Derivative of a Constant Function
Derivative of the Identity Function
Power Rule 1
The Sum Rule
The Product Rule
The Chain Rule
The Generalized Power Rule
Please select a rule from the menu


Physics KCD Previewer - Microsoft Internet Explorer
Compare your Solution to the Correct one:
Your Solution
Correct Solution
The Sum rule states,
Power Rule 1

Overall, we found a significant interaction effect using an ANCOVA with Post-test scores as the dependent variable, LA/EN and HI/LO as factors, and Pre-test and Middle-test scores as covariates F(1,30) = 7.47, $p < .05$, MSE= 7.41. In a post-hoc analysis using a Bonferroni test, the students in the Engaged High performing condition achieved significantly higher post-test scores than the students in the Engaged Low performing condition, $p < .05$. There was a marginal trend in favor of Lazy Low in comparison with Engaged Low $p < .1$. Lazy High was indistinguishable from the other conditions. Overall we did not find evidence that the errors contributed by the fake peer learner were harmful except in the case of Engaged peer learners.

Because the difference between Lazy Low and Engaged Low was marginal, we wanted to investigate further whether this effect was real or by chance. We checked first to see if the difference in impact between Lazy Low and Engaged Low could be explained by the difference in the distribution of labor that resulted as an effect of the confederate peer learner's behavior. For each student we recorded the ratio between that student's contributions to the problem solving and the total number of contributions. We refer to this variable as Labor. Pre-test score was reliably correlated with Labor across the students in all 4 experimental conditions (R-squared=.369, $p<.001$; N=36). We found a main effect of the LA/EN factor on Labor favoring Lazy over Engaged even with effect of Pre-test factored out, F(1,33)=22.45, $p<.001$, effect size 1.6 standard deviations). Although we instructed students to keep their division of labor 50%/50%, and although the median value was 48%, the standard deviation was 12%. So students did not stick rigidly to the 50%/50% division of labor, and the frequency of the student subjects' contributions was affected by the frequency of the confederate peer learner's contributions. If this explained the marginal difference in effectiveness between Lazy Low and Engaged Low, however, we would expect to see a correlation between Labor and learning. Nevertheless there was no correlation between Labor and Post-test with effect of Pre-test and Mid-test factored out.

Interestingly, taking students with below the median Pre-test scores as low ability students and others as high ability students, we found a trend that low ability students were more likely to let the Low performing peer learners take the lead than the High performing peer learners whereas the high ability students were more likely to trust the High performing peer learners than the Low performing peer learners. It's possible that low ability students were more likely to trust the low performing peer learners because they found them more confusing. The effect was only significant within the Engaged condition. Nevertheless, there was evidence that students did notice the errors contributed by the confederate peer learners. We informally observed a lot of discussion surrounding the errors. Taking average student turn length as an indicator of how much discussion beyond minimal answers was part of the conversation, we also found a marginal difference in turn length using an ANCOVA with HI/LO as a factor and turn length during tutoring as a covariate F(1, 28)=3.67, $p<.1$, MSE=2.4. Overall average turn length was shorter during the peer problem solving phase than during the tutoring phase. However, this pattern was less true of students working with Low performing (LO) confederate peer learners. A greater proportion of student in the HI condition had shorter turn lengths during the collaborative learning phase than during the tutoring phases than in the LO condition. This was true for 88% of the students in the HI condition and only 40% of the student in the LO condition, which was a significant difference in proportions according to a binary logistic regression ($P < .005$).

We suspected that the confederate peer learner might have contributed more errors overall in the Engaged condition (because of the difference in frequency of contribution), and that low ability students were exposed to many more errors in the Engaged Low condition, and that perhaps it was the number of errors that the student was exposed to during the peer problem solving phase that was having the real effect on student learning. To simplify the analysis, we introduced a new variable called ActualLazy/Engaged that was based on the division of labor (Labor) to more precisely estimate the number of errors students were exposed to. However when we replaced the LA/EN variable with the ActualLazyEngaged variable, the interaction effect observed previously became non-significant.

To further investigate our hypothesis about the number of errors students were exposed to, we examined the correlation between learning and on-line performance measures such as solutions submitted, correct solutions, and incorrect solutions. We found that the strongest predictor of student learning was the number of correct problems the pairs managed to submit during the problem solving phase (CorrectProb). We computed this with a linear regression between CorrectProb and Post-test score with effect of Pre-test score factored out. R-squared=.70, p<.001, N=36. There was a main effect of the HI/LO factor on the number of correct solutions contributed, with the effect of Pre-test and Mid-test scores used as covariates, F(1,30) = 49.1, p < .001, MSE=.93, effect size = 2.4 standard deviations. Students in both High performing conditions contributed significantly more correct solutions that students in either Low performing condition. Since there was a strong correlation between Mid-test score and correct problems contributed, we replaced mid-test with correct problems submitted as a covariate in the original ANCOVA with LA/EN and HI/LO as factors. We used Pretest score and Correct Problems submitted as covariates. While Pretest and CorrectProb submitted together explain about 71% of the variation in post test scores across our student population, we still found a significant crossover interaction effect explaining an additional 4% of the variance that provided some weak evidence that the errors contributed by the fake peer learners sometimes had a positive effect on student learning. F(1,30) = 4.96, p<.05, MSE=10.68. On the continuum between High and Low performing peer learners, students in the Lazy condition learned more when the peer learner contributed more errors, whereas the trend was the opposite with Engaged peer learners. This is consistent with findings reported about “troublemaker conflicts” reported in (Aimeur, Frasson, & Lalonde, 2001).

Conclusions and Current Directions

The results of this investigation contribute insights towards a detailed causal model of how environmental factors influence student perceptions, attitutudes, behavior, and learning. An understanding of where errors can be used strategically to stimulate cognitive conflict and student learning may enhance the effectiveness of existing well-established approaches to scaffolding in intelligent tutoring systems. Nevertheless, this is an issue that requires more investigation. Because the majority of the observed learning in this study is explained by correct problem solving, these results do not argue that errors play a large role in student learning relative to correct examples. The weakness of this effect might be explained by a paucity of what is referred to as “high level” explanation and help seeking behaviors found in our corpus of collaborative problem solving interactions (Webb et al., 2002). Webb et al. found, for example, that high ability students only benefited from their interactions with lower ability peers when their group engaged in high level explanation and help-seeking behaviors.

Acknowledgements
This work was funded by NSF SGER REC-0411483 and NSF/IERI REC-043779.

References

[1] Aimeur, E., Frasson, C., Lalonde, M. (2001). The Role of Conflicts in the Learning Process, *SIGCUE OUTLOOK* 27(2).

[2] Aleven V., Koedinger, K. R., & Popescu, O. (2003). A Tutorial Dialogue System to Support Self-Explanation: Evaluation and Open Questions. *Proceedings of the 11th International Conference on Artificial Intelligence in Education, AI-ED.*

[3] Anderson, J. R., Reder, L. M., & Simon, H. A. (1997). Situative versus cognitive perspectives: Form versus substance. *Educational Researcher*, 26(1), 18-21.

[4] Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. (2004). Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game The System". *Proceedings of ACM CHI 2004: Computer-Human Interaction*, 383-390.

[5] Bloom, B. S. (1984). The 2 Sigma Problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13, 4-16.

[6] Brown, J. S., Burton, R. R., & deKleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems* (pp. 227-282). New York: Academic Press.

[7] Carbonell, J. R. (1969). On man-computer interaction: a model and some related issues. *IEEE Transactions on Systems Science and Cybernetics* 5(1): 16-26.

[8] Chan, T-W., Baskin, A.B. (1988). Studying with the Prince: the computer as a learning companion. *The Proceedings of the International Conference on Intelligent Tutoring Systems*, Montreal, Canada, 194-201.

[9] Davis, J. M., Leelawong, K., Belynne, K., Bodenheimer, R., Biswas, G., Vye, N., & Bransford, J. (2003). Intelligent User Interface Design for Teachable Agent Systems. *Proceedings of the International Conference on Intelligent User Interfaces* (pp26-33).

[10] Dillenbourg, P. & Self, J. (1992). People Power: a human-computer collaborative learning system. In Gauthier, G. & McCalla, G. (eds.), *Intelligent Tutoring Systems. Lecture Notes in Computer Science 608,* Springer Verlag, 651-660.

[11] Evens, M. and Michael, J. (2003). One-on-One Tutoring by Humans and Machines, Lawrence Earlbaum and Associates.

[12] Graesser, A. C., Bowers, C. A., Hacker, D.J., & Person, N. K. (1998). An anatomy of naturalistic tutoring. In K. Hogan & M. Pressley (Eds.), *Scaffolding of instruction*. Brookline Books.

[13] Graesser, A., VanLehn, K., the TRG, & the NLT (2002). *Why2 Report: Evaluation of Why/Atlas, Why/AutoTutor, and Accomplished Human Tutors on Learning Gains for Qualitative Physics Problems and Explanations*, LRDC Tech Report, University of Pittsburgh.

[14] Hietala, P. & Niemirepo, T. (1998). The Competence of Learning Companion Agents. International *Journal of Artificial Intelligence in Education*, 9, pp178-192.

[15] Leelawong, K., Davis, J., Vye, N., Biswas, G. (2002). The Effects of Feedback in Supporting Learning by Teaching in a Teachable Agent Environment, *Proccedings of ICLS'02*.

[16] Palthepu, S., Greer, J. and McCalla, G. (1991). Learning by teaching. In *The Proceedings of the International Conference of Learning Sciences*. AACE.

[17] Rosé, C. P., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., & Weinstein, A. (2001). Interactive Conceptual Tutoring in Atlas-Andes, In J. D. Moore, C. L. Redfield, & W. L. Johnson (Eds.), *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future, Proceedings of AI-ED 2001* (pp. 256-266). (2001) Amsterdam, IOS Press.

[18] Sharan, S. (1980). Cooperative Learning in Small Groups: recent methods and effects on achievement, attitudes, and ethnic relations. *Review of Educational Research* 50:241-271.

[19] Stevens, A., & Collins, A. (1977). The Goal Structure of a Socratic Tutor, In *Proceedings of the National ACM Conference., Association for Cmputing Machinery,* New York

[20] Vizcaino, A. & Du Boulay, B. (2002). Using a Simulated Student to Repair Difficulties in Collaborative Learning, In *Proceedings of ICCE'2002*, New Zealand. IEEE, 2002.

[21] Webb, N., Nemer, K., & Zuniga, S. (2002). Short Circuits or Superconductors? Effects of Group Composition on High-Achieving Students' Science Assessment Performance, *American Educational Research Journal*, 39, 4, pp 943-989.

Generating Structured Explanations of System Behaviour Using Qualitative Simulations

Anders BOUWER and Bert BREDEWEG
University of Amsterdam, Human-Computer Studies Lab
The Netherlands
Email: {bouwer,bredeweg}@science.uva.nl

Abstract. This paper presents an approach to generate structured explanations of system behaviour based on qualitative simulations. This has been implemented in WiziGarp, a domain-independent interactive learning environment. The main issue addressed here is how to manage the complexity of a simulation in order to generate adequate explanations. These are presented to the user in the form of different kinds of diagrams, accompanied by explantory dialogue.

1. Introduction

In many domains, such as physics, biology, chemistry, and ecology, learning about dynamic phenomena is essential. Learners must be able to recognize, predict and explain the behaviour of systems in relation to their structure [6]. This requires having an adequate conceptual model of behaviour, which can relate a particular state of the system to previous states (for explanation) or future states (for prediction). Qualitative simulations explicitly represent the kinds of knowledge that can support learners in building their own conceptual model of dynamic phenomena [12]. The GARP framework [3], which is used in this work, includes knowledge about entities, relationships, quantities, and causal and mathematical dependencies. This knowledge is organized in scenarios, model fragments, and transition rules, and is used to generate a state-transition graph of all possible behaviours of the system.

The main problem in using qualitative simulations for educational purposes is that due to the amount of detail included, a simulation can be very complex, *i.e.*, containing a large number of states and transitions, and a large amount of information within each state. Some approaches to this problem try to simplify the model and the resulting simulation beforehand, based on specific questions that the simulation should address [5, 10]. However, this is often not possible, because it is in general unknown in advance which questions the simulation should answer.

Therefore, the approach described in this paper takes a simulation as a given and performs aggregation afterwards. This enables automatic generation of the questions that the simulation can answer, and filtering of the most important information, while still allowing full access to the details of the original simulation.

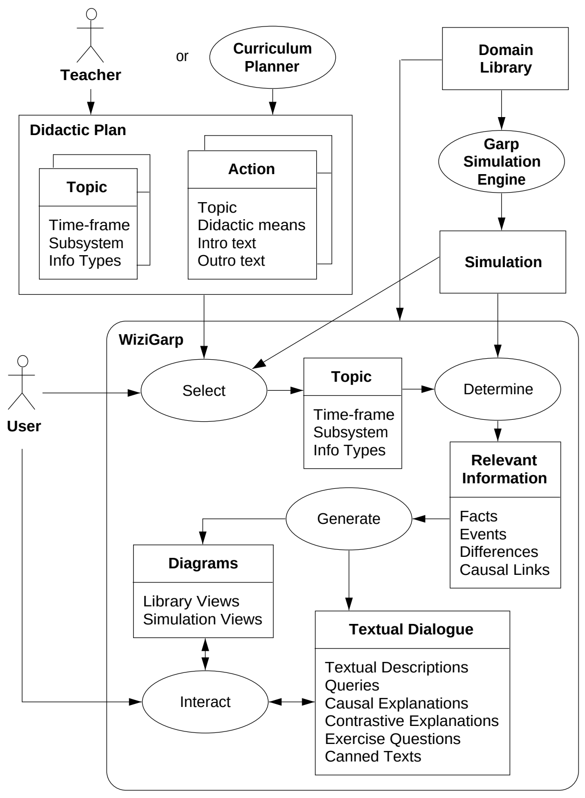


Figure 1. An overview of WiziGarp's system architecture, showing the main data and control flow

2. WiziGarp and Domain Example: Cerrado Ecology

The context of this work is the development of an interactive learning environment called WiziGarp, which extends its predecessor VisiGarp, described in previous work [2]. In figure 1, an overview of WiziGarp's system architecture is shown, including the main data and control flow. Based on a generic domain library and a particular simulation, WiziGarp generates diagrams and textual explanatory dialogue for specific topics, which are either selected directly by the learner, or specified in a didactic plan, provided by a human teacher or a curriculum planner.

The test domain used in the remainder of this paper is the ecology of the Brazilian Cerrado vegetation, as modelled qualitatively in the GARP framework [11]. The model

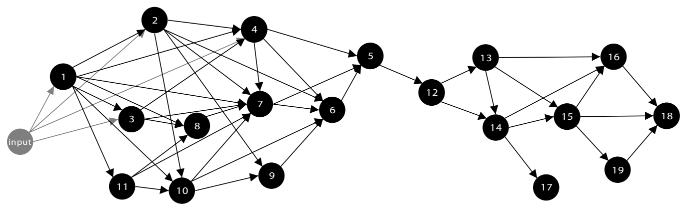


Figure 2. The original state-transition graph for the CSH simulation

Level	Contents
0. GARP	Output from the simulator
1. System state	Current state of the system
2. Local event	Events at time point/interval
3. Path segment	Necessary behaviour (sequence w.o. branching point)
4. Path	Possible behaviour (may include branching points)
5. Global	Alternative behaviours (multiple paths)

Table 1. Levels of aggregation

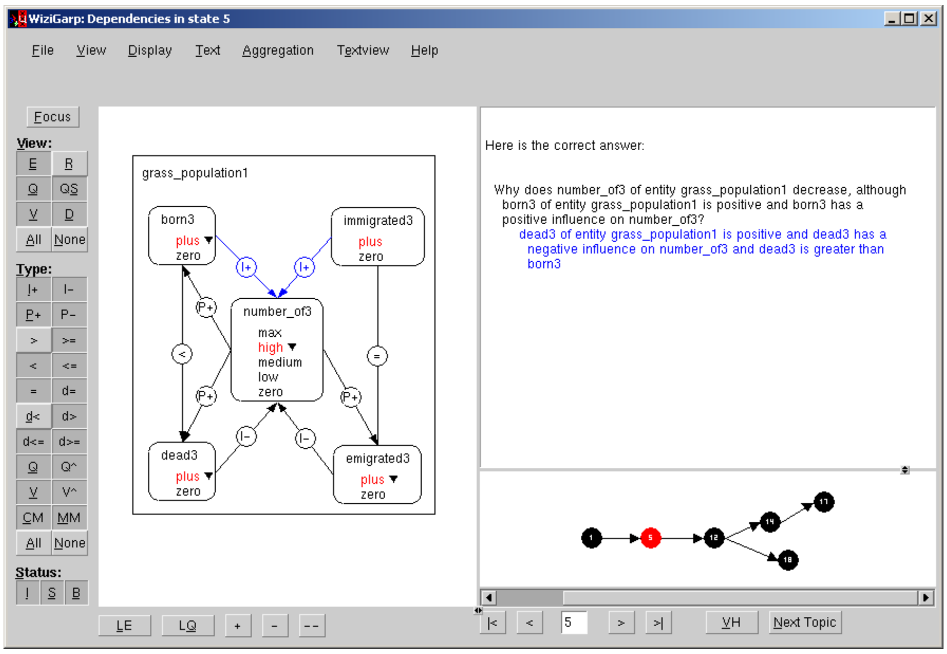
includes a classification of vegetation types, ranging from Campo Limpo, consisting of mainly grassland, to Cerradão, a vegetational state with mainly trees and shrubs, and (almost) no grass population. One of the simulation scenarios models the Cerrado Succession Hypothesis (CSH), which suggests that a Campo Limpo may develop towards a Cerradão under the right circumstances, in terms of natural conditions and human actions, such as fire management. The state-transition graph for this simulation is shown in figure 2. it contains 19 states and 47 transitions, which can be combined into 869 different paths. Each of the states contains on average 72 entities and relations, 32 quantities, 99 dependencies, and 50 model fragments. This combination leads to a total number of 4800 facts contained in the simulation. It would be practically impossible to communicate such a high number of facts to a user, and presenting them in chronological order poses a problem because there are so many different paths possible.

The aggregation techniques described in the next section are therefore used to reduce the amount of information to be communicated, both in terms of the number of states and the amount of information within states. The output of the aggregation methods is then utilized by WiziGarp to generate structured explanations. The goal of these explanations is to provide insight in the simulation results and the causal mechanisms involved.

3. Aggregation of Qualitative Simulations

The approach to aggregation presented in this paper is to organize the information in a qualitative simulation in ways which support the simplification, interpretation and selection of interesting information. In order to do this, *levels of aggregation* are introduced, which correspond to different views on system behaviour. The five levels are shown in table 1, together with their mapping to the corresponding notions in the state-transition graph of a simulation.

For each of these levels, aggregation techniques have been implemented which reduce the amount of information to be communicated. These techniques are based on



Here is the correct answer:

Why does number_of3 of entity grass_population1 decrease, although born3 of entity grass_population1 is positive and born3 has a positive influence on number_of3?
dead3 of entity grass_population1 is positive and dead3 has a negative influence on number_of3 and dead3 is greater than born3

Figure 3. A screenshot from WiziGarp: dependencies for the grass population and an exercise question

four generic principles: selection (focusing on certain elements), chunking (combining a sequence of information elements into a new element), generalization (abstracting from minor differences between information elements to form a new element), and grouping (adding a label to a group of similar elements).

On the system state level, the status of causal dependencies is analyzed to arrive at a *classification of causal effects*, as inactive, submissive, balanced, or effective. This allows grouping and selection of those dependencies which have an actual effect, while discarding dependencies whose effect does not contribute to the outcome of the simulation. The classification refines work by De Koning et al. [4], who distinguish only between submissive and effective dependencies, and work by Mallory [9], who attributes similar labels directly to quantities instead of to their effects. Compared to both these efforts, our classification enables better explanations of why a causal influence may not have the expected effect, especially in cases where a particular quantity is involved in multiple (possibly competing) dependencies. Figure 3 shows an example screenshot from WiziGarp with a subset of dependencies for the grass population in the Cerrado, including submissive dependencies (the positive influences from born3 and immigrated3), as well as effective ones (the negative influences I- from dead3 and emigrated3 and the positive proportionalities P+, which further propagate the decrease of number_of3).

On the local event level, *recognition of events* is performed. Information from adjacent states is selected and combined to form larger chunks of information, specifying meaningful events of various types, such as value and derivative events (e.g., Q_x starts to increase), inequality events (e.g., Q_x becomes larger than Q_y), causal effect events (e.g., the influence from Q_x on Q_y becomes inactive), and model fragment events (e.g., the model fragment for a particular process or situation becomes active). To be able to present an appropriate number of events, in an orderly fashion, WiziGarp allows selection and grouping of events by topic. As shown in figure 1, a topic contains a specific time-frame (particular states or paths), subsystem (the entities and quantities of interest), and information types (the kinds of events of interest).

On the next two levels, the state-transition graph is divided into path segments or paths, respectively. A path segment is a state sequence without (outgoing) branching points. Conceptually speaking, a path segment specifies behaviour that necessarily takes place. A path, on the other hand, may extend beyond a branching point. When a path contains branching, the events after the branching point do not necessarily occur. Paths are also analyzed for the existence of cycles to recognize repetitive behaviour. On the path segment level and path level, additional value and derivative events are recognized on this level by selection and chunking of lower level events (e.g., the highest value of Q_x that is reached in the path (segment) P is V, or Q_x fluctuates between V_1 and V_2, respectively).

Finally, on the global level, *transitive reduction* decreases the number of transitions (selection), and *aggregation of alternative orderings* decreases both the number of transitions and states in the state-transition graph by generalizing over reuniting branches. Transitive reduction, as known from graph theory, abstracts from all transitions T ($= S_x \rightarrow S_y$) for which holds that there is a path P from S_x to S_y which does not contain transition T, with the extra condition that P contains the same events as T. Aggregation of alternative orderings abstracts paths which divert and reunite, if they include the same events, albeit in a different order. This technique is defined by the following algorithm:

Algorithm 1. Aggregation of alternative orderings in the state-transition graph:

Find a group of paths P_1 to P_n with the same begin-point (S_x) and end-point (S_y), for which holds: P_1 to P_n contain the same events, or events which can be abstracted into the same higher level events, and do the following:

1. Add a shortcut edge from S_x to S_y, to represent an aggregated transition, containing all (aggregated) events (of all, or selected event types) occurring in paths P_1 to P_n.
2. Delete every edge from the original paths P_1 to P_n, unless:
   (a) the edge appears *after* an *incoming* branching point, OR
   (b) the edge appears *before* an *outgoing* branching point.
3. Delete states which have no incoming and outgoing edges anymore.

Repeat this process until no more alternative paths can be found.

Both of these techniques are based on the idea that it is useful to abstract from the temporal order if the general story is the same, and it does not matter for later developments in the simulation. After transitive reduction (see figure 4(a)), the state-transition graph still contains 19 states, but the number of transitions is reduced from 47 to 26, and the number of paths from 869 to 24, compared to the original state-transition graph. After

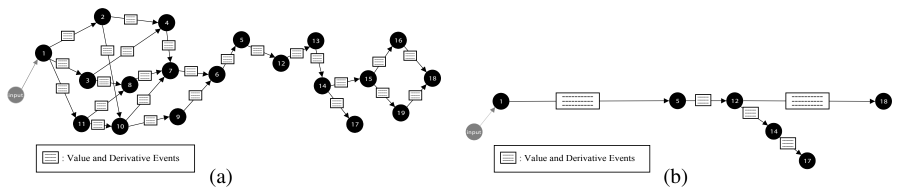


Figure 4. The state-transition graph after transitive reduction (a), and after aggregation of alternative orderings (b)

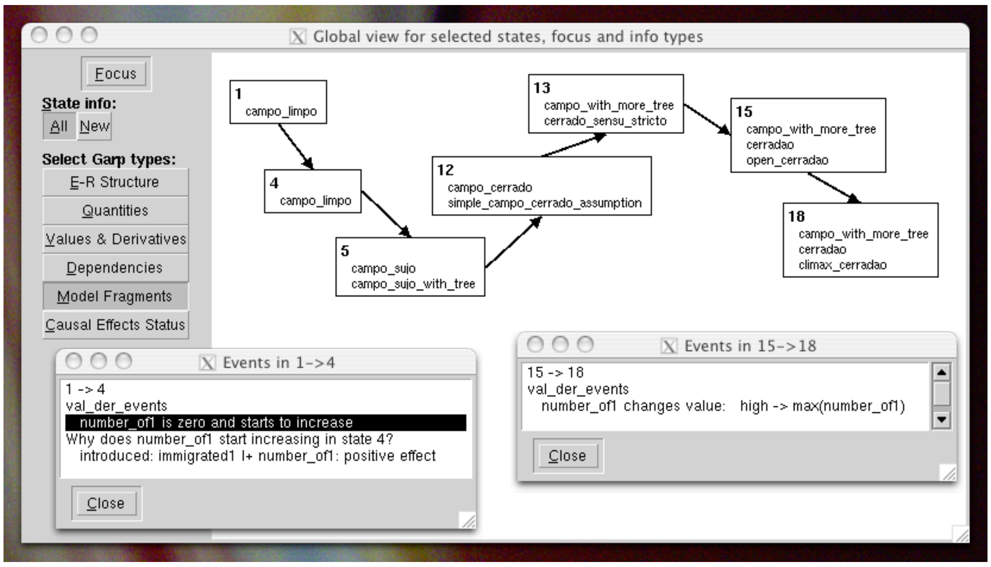
aggregation of alternative orderings, the graph is reduced to 6 states, 6 transitions, and only 2 paths (see figure 4(b)). It is clear that without these forms of reduction, it would be impractical to discuss what happens in all paths of the simulation.

4. Generating Interactive Explanations

The generation of explanations is done based on a didactic plan. The didactic plan specifies which topics should be presented in what form (*i.e.*, the didactic means: diagrams, textual descriptions, causal explanations, contrastive explanations, queries or exercise questions), as well as the desired level of aggregation. In the CSH domain, the sequence of topics in the didactic plan starts with the entity-relationship structure, followed by the development of values and derivatives. Then, the dependencies are treated, first for individual entities (the grass, shrub and tree populations), and later for interactions between multiple entities (e.g., the effect of the manager on the populations, mediated by vegetational factors such as humidity and soil temperature).

The different didactic means can be used to support didactic styles from system-initiated tutoring to student-initiated exploration. WiziGarp can take the initiative by asking exercise questions that are automatically generated by a separate module called QUAGS, which uses heuristics, or specific input from the didactic plan [8]. An example was shown already in figure 3, with a question about the grass population. WiziGarp incorporates heuristics to determine the most interesting quantities, based on which quantities are involved in the highest number of events for the selected time-frame, such as quantity number_of3. In many cases, it is useful to hide submissive (and inactive) dependencies, as they do not contribute to, and may therefore distract attention from the actual behaviour. In this case, however, the four basic processes in population ecology are an important topic to learn about. Therefore, the status of their effects is taken into account in the exercise question.

Figure 5 shows how a learner can also take the initiative and ask for specific information to be answered by WiziGarp. The diagram in the figure shows an overview of the main developments for a particular path in the simulation. In order to do this, events on the local level and path (segment) level are determined for the quantities of interest. Within the boxes depicting states, the model fragments are displayed which specify the vegetation types according to the CSH. These model fragments are defined in terms of the values of three quantities: $number\_of1(trees)$, $number\_of2(grass)$, and $number\_of3(grass)$. For example, the model fragment *climax_cerrado* is associated with a maximum number of trees. The learner has selected transitions $1 \rightarrow 4$ and $15 \rightarrow 18$ to see which value and derivative events occur. The learner can ask queries about a particular event by selecting it and choosing a query from a popup menu that


Global view for selected states, focus and info types
Focus
State info:
All
New
Select Garp types:
E-R Structure
Quantities
Values & Derivatives
Dependencies
Model Fragments
Causal Effects Status
1 campo_limpo
4 campo_limpo
5 campo_sujo campo_sujo_with_tree
12 campo_cerrado simple_campo_cerrado_assumption
13 campo_with_more_tree cerrado_sensu_stricto
15 campo_with_more_tree cerradao open_cerradao
18 campo_with_more_tree cerradao climax_cerradao
Events in 1->4
Close
Events in 15->18
Close

```
1 -> 4
val_der_events
   number_of1 is zero and starts to increase
Why does number_of1 start increasing in state 4?
   introduced: immigrated1 I+ number_of1: positive effect
```

```
15 -> 18
val_der_events
   number_of1 changes value:   high -> max(number_of1)
```

Figure 5. An overview of the main developments in terms of model fragments for the CSH

arises. Here, the learner asks why the number of trees has started to increase, which is causally explained by the introduction of the effect of immigration.

In addition, contrastive explanations can be generated which list the differences between states, e.g., the end states 17 and 18: only in the latter case, the tree population reaches its maximum size, and the grass population starts to increase again due to immigration, after dying out first.

5. Conclusion

The main research goal addressed in this paper was how to deal with the complexity of qualitative simulations in order to generate structured explanations of system behaviour. A set of aggregation techniques has been implemented which select, chunk, generalize, or group information elements, so that the result contains less states and transitions, as well as less information within them. This was done by focusing on events of various kinds, at different levels of abstraction. These aggregation techniques have been incorporated into a prototype interactive learning environment called WiziGarp. Based on the output of the aggregation mechanisms, explanatory dialogue is generated including several didactic means: textual descriptions, causal explanations, contrastive explanations, queries, and exercise questions, complemented by diagrammatic representations of simulation results and the underlying model. An example of explanatory interaction has been presented which shows that for the domain of Brazilian Cerrado ecology, the aggregation mechanisms provide a large reduction in the amount of information. This makes it feasible to communicate such information to learners, which is considered especially useful for students of ecology and future decision makers in the field of sustainable development. Our approach is generic and has also been tested on other domains, such as physics and biology. Several parts of the WiziGarp architecture have been evaluated by potential users and domain experts, such as the the question generation module [8], and the diagrammatic representations [1]. The results of these evaluation studies are encouraging and have led to several improvements from predecessor VisiGarp (which included only the diagrammatic representations, without aggregation mechanisms and textual explanatory dialogue) to the design of WiziGarp.

Compared to related work that addresses explanations for simulations [7, 9], WiziGarp encompasses a more extensive taxonomy of events, more flexible aggregation mechanisms, and a richer set of didactic means. Future work will address an automated reactive curriculum planner, based on learner answers to exercise questions, thereby further increasing the level of interactivity to support learning about dynamic phenomena.

References

[1] A. Bouwer. *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*. PhD thesis, University of Amsterdam, to appear.

[2] A. Bouwer and B. Bredeweg. Visigarp: Graphical representation of qualitative simulation models. In J.D. Moore, C.L. Redfield, and J.L. Johnson, editors, *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*, pages 294–305, Ohmsha, Japan, Osaka, 2001. IOS-Press.

[3] B. Bredeweg. *Expertise in Qualitative Prediction of Behaviour*. PhD thesis, University of Amsterdam, Amsterdam, 1992.

[4] K. de Koning, B. Bredeweg, J. Breuker, and B. Wielinga. Model-based reasoning about learner behaviour. *Artificial Intelligence*, 117:173–229, 2000.

[5] B. C. Falkenhainer and K. D. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

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

[7] K. D. Forbus, P. B. Whalley, J. O. Everett, L. Ureel, M. Brokowski, J. Baher, and S. E. Kuehne. Cyclepad: An articulate virtual laboratory for engineering thermodynamics. *Artificial Intelligence*, 114(1/2):297–347, 1999.

[8] F. Goddijn, A. Bouwer, and B. Bredeweg. Automatically generating tutoring questions for qualitative simulations. In P. Salles and B. Bredeweg, editors, *Proceedings of QR'03, the 17th Int. workshop on Qualitative Reasoning*, pages 87–94, Brasilia, Brazil, 20-22 Aug. 2003.

[9] R. S. Mallory. *Tools for Explaining Complex Qualitative Simulations*. PhD thesis, Department of Computer Sciences, University of Texas at Austin, 1998.

[10] J. Rickel and B. W. Porter. Automated modeling of complex systems to answer prediction questions. *Artificial Intelligence*, 93:201–260, 1997.

[11] P. Salles and B. Bredeweg. Constructing progressive learning routes through qualitative simulation models in ecology. In G. Biswas, editor, *Proceedings of QR'01, the 15th Int. workshop on Qualitative Reasoning*, pages 82–89, San Antonio, Texas, 17-19 May 2001.

[12] R. Winkels and B. Bredeweg. Qualitative models in interactive learning environments. *Interactive Learning Environment*, 5:1–134, 1998. Editorial special issue.

Initial results and mixed directions for Research Methods Tutor [1]

Peter Wiemer-Hastings [a,2], Elizabeth Arnott [b] and David Allbritton [b]
[a] *DePaul University School of Computer Science, Telecommunications and Information Systems*
[b] *DePaul University Department of Psychology*

Abstract. RMT (Research Methods Tutor) is a dialog-based tutoring system currently being used in conjunction with university-level courses on research methods in psychology. RMT has a web-based interface and uses a talking head to to present its dialog acts to the student. The student types in unconstrained natural language responses. Although RMT currently focuses on natural language-based interaction with students, the use of the talking head brings in other modalities of interaction which must be integrated with the textual "message". This paper describes the basic architecture and approach of the RMT system, and its evaluation in the context of recent research methods classes. Our experiments compared the agent-based tutor to a text-only version of the system, and compared tutoring to a computer-aided instruction control. Due to technical difficulties with the agent software and some compliance issues with the students, we got no significant results that validated the usefulness of the system as a learning aid. We did however see consistent trends of additional learning in conjunction with the use of the tutor that warrant further investigation. This paper concludes by describing our current efforts for integrating graphical visual aids with dialog-based tutoring.

Keywords. Dialog-based tutoring, Interactive pedagogical agent / talking head, Graphics and text

1. Introduction

In the past decade, advances in natural language processing techniques have made it possible to create intelligent tutoring systems which interact with students more and more like human tutors do: via dialog about some content material. In some cases this interaction is appropriately entirely linguistic. In most cases, however, tutor-student interactions can and should include other types of shared information. This paper describes Research Methods Tutor (RMT), a dialog-based intelligent tutoring system for research methods in psychology. RMT is intended as an adjunct to a college-level course to be used by the student at their convenience to strengthen their understanding of the concepts discussed in class.

[1]This work was supported by a grant to the first author from DePaul University's Quality of Instruction Council.

[2]Correspondence to: Peter Wiemer-Hastings, School of Computer Science, Telecommunications and Information Systems, 243 South Wabash Avenue, Chicago, IL 60604, United States. Tel.: +1 312 362 5736; Fax: +1 312 362 6116; E-mail: peterwh@cs.depaul.edu.

RMT is embodied by an animated agent which uses text synthesis to present its questions, feedback, and other dialog moves. The tutor can also operate in text-only mode where its utterances are printed on the screen. The latter approach is much simpler from a technological viewpoint. On the other hand, there is evidence that students may pay scant attention to textual information in an ITS setting [6]. Furthermore, if graphics or other visual aids are presented simultaneously with explanatory text, the student's cognitive and perceptual resources for processing visual input may become overburdened [2]. In addition, the use of a "talking head" provides the possibility for utilizing another type of "mixed language", namely the intonation and facial gestures which human tutors often use to give graded feedback.

The RMT project has 3 major goals:

1. to explore the requirements for and benefits of a web-based tutoring system as an adjunct to classroom education,
2. to learn more about tutoring in an abstract, relatively informal domain as compared to science or math-oriented tutoring, and
3. to serve as a workbench for evaluating different aspects of dialog-based tutoring, for example how the use of a talking head might help students learn when compared to text alone.

This paper gives a high-level description of the RMT architecture and its approach to tutoring. Next, we present the results of our initial evaluations of the system with students in research methods courses at DePaul University. Finally, we discuss the directions which we are currently exploring for incorporating mixed-language explanations to help improve student learning.

2. RMT Basics

Psychology students at DePaul, as at most universities, are required to take one or more courses in research methods. These courses address the basic methodology required to do experimental psychology, including topics like variables, reliability and validity, different types of experimental designs, and ethics. Although the courses normally include many examples of aspects of experimental design, it is not the specific examples that the students must learn, but rather the processes for creating successful designs. This level of abstraction makes the courses especially difficult. When students interact with RMT, they have the opportunity to discuss additional examples and get direct feedback on their understanding and use of the relevant concepts. From a constructivist viewpoint, this active processing should increase the number of connections that they can make to related material and deepen their understanding.

RMT is a web-based system. Students log in to the system at their own convenience, typically during a window of time set by the instructor for each topic module, and typically from their own computers at home. The agent is implemented using Microsoft Agent software with its accompanying text-to-speech engine. The tutoring sessions start when the student has chosen one of the available topics. The tutor is normally "in control" of the dialog. The tutor asks questions, and the student types in his or her responses. The tutor evaluates each response using Latent Semantic Analysis [4,7] by comparing it to a set of one or more expected answers. The tutor then gives positive or negative feed-

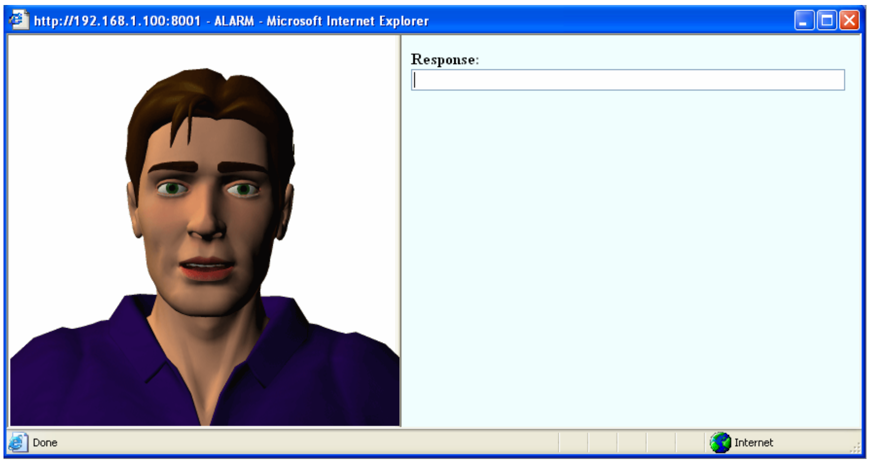
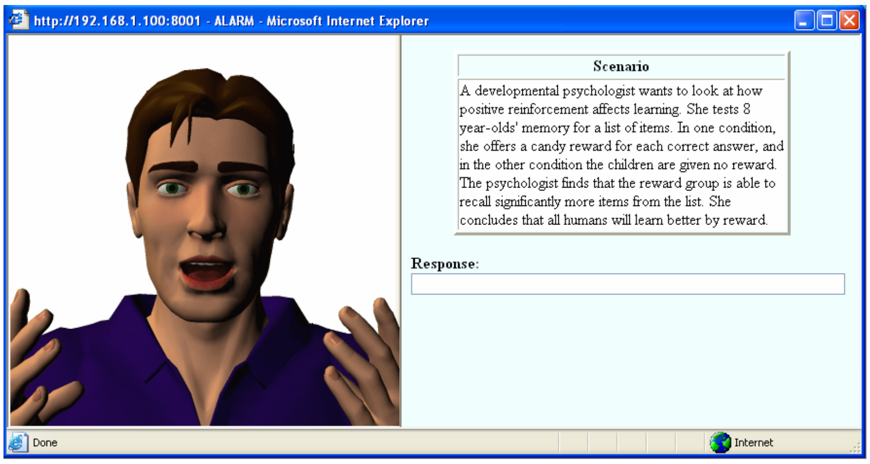

http://192.168.1.100:8001 - ALARM - Microsoft Internet Explorer
Response:
Done
Internet

Figure 1. The basic RMT interface

back and either tries to get the student to add more information in response to the current question, or summarizes and moves on to a new question. RMT uses a transition network to control its dialog decisions, including responding to questions and certain requests or commands (e.g. asking the tutor to repeat its last utterance). All tutor and student utterances are logged in a database along with the tutor's evaluations of student response quality. Figure 1 shows a screenshot of the (agent-based) tutoring interface. The interface was intentionally kept as simple as possible, avoiding anything that might distract the student from the tutoring interaction.

Currently RMT uses entirely linguistic interactions with the student. For most tutoring items, the tutor simply asks a question, evaluates the the student's response, and follows with another question or prompt to keep the dialog moving. Even within this limited paradigm, there is an aspect of mixed language. As human tutors do [5], RMT avoids the use of direct negative feedback which might cause the student to become embarrassed or "lose face" and subsequently reduce their level of participation in the dialog. Instead RMT uses intonation and facial and hand gestures to suggest that something is not quite right without explicitly telling the student that they're wrong. Thus the use of an animated agent automatically brings in multiple modes of communication. In RMT, these modes are coordinated by its feedback mechanism which is invoked with an utterance and an affective stance like "positive" or "confused." The feedback mechanism will then choose an intonation pattern and animation to express this affect. A random component in the feedback mechanism keeps the agent from acting too "robot-like".

RMT has three different levels of tutorial material: conceptual, analytic, and synthetic [1] which roughly correspond to the student learning the basic concepts, how to apply them to analyze example experimental designs, and how to create new experimental designs. Many of the analytic and synthetic items include a fairly detailed description of a particular experimental scenario which is then referred to over the course of an extended discussion. It would be unreasonable to expect the students to maintain the details of these examples in their working memory while the discussion goes on. To provide an external memory device, RMT always presents these scenarios on the screen as shown in Figure 2 during the related dialog. The tutor refers to the scenario when asking the initial


http://192.168.1.100:8001 - ALARM - Microsoft Internet Explorer
Done
Internet

Scenario

A developmental psychologist wants to look at how positive reinforcement affects learning. She tests 8 year-olds' memory for a list of items. In one condition, she offers a candy reward for each correct answer, and in the other condition the children are given no reward. The psychologist finds that the reward group is able to recall significantly more items from the list. She concludes that all humans will learn better by reward.

Response:

Figure 2. RMT with a scenario

question and gives the student the opportunity to read the scenario while coming up with their answer.

This relatively simple architecture allows RMT to engage students in extended conversations about a wide range of topics in research methods. We have currently developed modules for five topics: validity, reliability, variables, experimental design, and ethics. The next section discusses our initial evaluations of the RMT in conjunction with research methods courses at DePaul.

3. Evaluation

The motivation behind our goal of integrating RMT with classroom education was to enable us to fill a gap in the current understanding of dialog-based tutoring. Previously such systems have been evaluated in the context of laboratory experiments where participants take a pre-test, use the system intensively for a significant amount of time, then take a post-test [3, for example]. This protocol certainly highlights the potential effectiveness of these systems, but may be lacking in external validity. The fundamental question is whether such systems can help students learn on an ongoing basis. Our “real-world”, class-linked evaluation of an ITS is challenging since the students are using the tutor for significantly less time than they are sitting in class throughout the course of a term. Furthermore, as we describe below, other aspects of the real world have complicated our evaluations. Nevertheless, our results were encouraging if not conclusive.

Pilot data from two classroom studies of the current version of RMT supported the feasibility and usefulness of our approach. During the Winter and Spring quarters of 2004, 101 students from multiple sections of Research Methods courses at DePaul University volunteered to use the tutor throughout the quarter for extra credit. A few of them actually did use the tutor for one or two sessions. One of our main research questions focussed on whether the use of an animated agent helped or hurt student learning when compared to the text-only version of the tutoring system. Unfortunately, because the agent-based modules required the installation of additional software, very few students actually completed these modules. Those who were able to install the software were greeted by an agent with no animations (except for blinking and lip movements) and a rather high, squeaky voice. Thus, we were not especially surprised when the agent version failed to help students learn the material.

In the Fall quarter of 2004, 28 students in a single section of Research Methods II (the second of the two-course methods sequence taken by all psychology majors at DePaul) who were required to use the tutor as a course assignment consented for their data to be analyzed for research purposes as well. The architecture of the system had been extended to support automatic and dynamic assignment of students to different conditions for the different modules. This second experiment tested the following hypotheses: First, that the RMT system would lead to greater learning gains than a computer assisted instruction (CAI) control[1]; and second, that an animated agent would be superior to text-only presentation.

Across the three classes in our sample (one section of Research Methods I, and two sections of Research Methods II) a total of 43 students completed both the pre-test and post-test and were thus included in the analysis (18 from RM I, and 25 from RM II). Of these, 15 were unable to install the agent software at all, leaving a relatively small number of observations per condition.

The design was a 2x2 within-subjects factorial, with presentation (text-only vs. agent) and instruction type (RMT tutor vs. CAI) as the independent variables. For each student, one module was assigned to each of the four conditions. Because there were five modules and four conditions, one of the conditions was assigned to two modules for each subject, with the pairing of modules and conditions counterbalanced across groups of four subjects.

Unfortunately, many subjects failed to complete one or more modules, with the result that the design was no longer counterbalanced across modules. We therefore analyzed each module separately, with pre-test to post-test gain score for proportion correct on questions related to that module as the primary dependent measure, and pre-test score included as a covariate. The design of the analysis of each module was therefore a 2x2 ANCOVA, with both presentation and instruction type as between-subject factors. Only students who had installed the agent software and had completed at least one session on that module were included in each analysis (N = 12, 12, 26, 25, and 24 for the validity, reliability, variables, experimental design, and ethics modules respectively). There were no reliable main effects or interactions in any of these analyses, probably due to the low Ns and variability associated with technical problems with the agent software reported by students. The only significant effect in the analyses was that of the pre-test covariate — not surprisingly, lower pretest scores were generally associated with higher gain scores.

Although one of our aims had been to compare the effectiveness of the agent to text-only presentation, our data was uninformative on this point, revealing no significant main effect of presentation type in any analysis, and no consistent qualitative pattern (three modules had higher covariate-adjusted mean gain scores for text-only presentation and two for agent presentation). Because of the lack of presentation effects, and the fact that over a third of our sample failed to install the agent software at all, we then collapsed across presentation type and re-analyzed the data, focusing instead on instruction condition.

[1] In the CAI control condition, students were presented with a series of static texts (in either text-only form or “read” by the animated agent) which covered the same general content material as the tutoring sessions, and they were given a multiple choice test at the end of each module to ensure that they attended to the text.

	None	CAI	Tutor
Validity	.05	.08	.22
Reliability	.07	.05	.13
Variables	.13	.08	.12
ExptDes	.00	.09	.13
Ethics	.08	.16	.19

Table 1. Gain scores by Module and Condition

In the re-analysis, we focused on two questions: (1) Is there any evidence that RMT produces greater learning gains than the CAI, or that either produces greater gains than those found for students who did not complete that module at all (none). (2) Does aptitude (as measured by overall pre-test scores, not separated by module) interact with instruction type? Our intuition based on student comments was that high-aptitude students may learn more effectively from the CAI, which allows them to control the pacing and presentation themselves, while lower-aptitude students may learn better from the tutor, which provides more active, tutor-directed instruction.

We re-analyzed the gain scores from all 43 students for each module in separate one-way ANCOVAs, with instruction condition (CAI, RMT, or none) as the between-subjects factor and module pretest score as the covariate. There was no significant effect of instruction in any of the analyses; again the only significant effects were those of the pretest covariate. Qualitatively, however, a fairly consistent pattern emerged. The mean covariate-adjusted gain score for the RMT condition was higher than that of the CAI condition for all five modules ($p < .05$ by a sign test) and higher than the baseline "none" condition for four of the five modules. The CAI condition was not consistently above baseline, however, with only two modules having a higher mean gain score for CAI than "none." Table 1 lists the average gain scores for the five modules by instruction condition.

This is consistent with the qualitative pattern of the overall unadjusted mean gain scores. Averaging across all modules, the mean gain scores were .10 for CAI, .15 for RMT, and .06 for uncompleted modules. Thus we have a trend that suggests that the use of RMT increases learning, although no solid statistical evidence from our limited sample.

To test our hypothesis of an aptitude by instruction-type interaction, we entered the module gain scores into separate 2x2 ANCOVAs, with instruction condition (CAI vs RMT) and aptitude (low vs. high, as defined by a median split of overall pretest scores) as between-subjects factors, and module-specific pretest score as the covariate. (Uncompleted modules were excluded from these analyses). The predicted aptitude by instruction interaction was not confirmed. There were no significant main effects or interactions in any of the analyses, with the only significant effects being those of the module pretest covariates (significant in 4 of the 5 analyses). Qualitative examination of the mean gain scores revealed no consistent pattern of interaction either. To make sure the lack of interaction was not simply an artifact of reducing the variability in the aptitude measure by making it categorical, we also conducted regression analyses with aptitude (overall pretest score), instruction type, and the aptitude*instruction type interaction as predictors. No significant interactions were found in any of these analyses either. Thus we have no evidence at present to support our hypothesis that the RMT tutor is more effective for low-aptitude students, compared to a CAI.

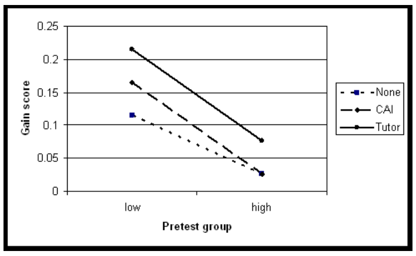


Figure 3. Average gain by instruction type and aptitude

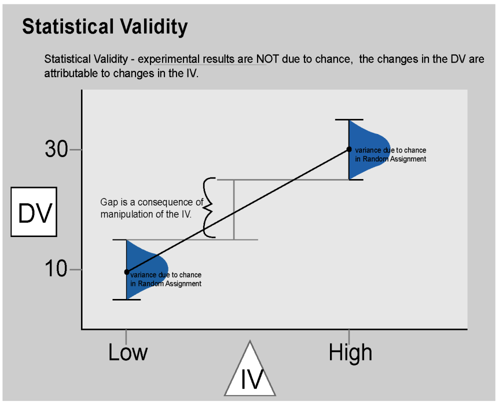
Despite the fact that the differences were not significant, the data trends did come out in the right direction. Figure 3 shows the differences in gain scores for the low and high pretest groups for the different types of instruction. In this chart, the average gain scores for the modules which students did not complete (i.e. the learning gain due to the coursework alone) were also included for comparison and are marked as "None." Again, we make no claims based on this outcome other than that this question of an aptitude by treatment interaction in RMT warrants further study.

4. Mixed Language Explanations

Research on e-learning systems shows that including graphics with textual explanations can produce a cognitive synergy in the learner that leads to more effective processing of the material and deeper understanding [2]. Unfortunately in an abstract domain like research methods, it is relatively easy to create graphics that illustrate a particular example (e.g. participants in an experiment drinking coffee before taking a test), but such graphics will do little to improve the student's far transfer. It is much more difficult to create graphical materials that illustrate the important concepts related to the processes, and there is still the question of how much they will help the students.

Figure 4 shows one graphic that we developed to explain the concept of statistical validity, i.e. that the changes in the dependent variable in an experiment are caused by changes in the independent variable, and not due to chance. The graphic includes some aspects that should be familiar to the students, like a basic 2-dimensional graph and frequency distributions. We still don't know, however, if such a graphic is too abstract to be beneficial to the students. Another issue is the method of presentation of the graphic. Should we start with a blank slate and add components as the discussion progresses, or should we present the entire graphic and center the discussion around getting the student to understand it? A further complication is related to the use of talking head along with the graphics. The animated head might distract the student from attending to the graphic. A solution might be to either have the head disappear or wander off-screen, or to have it look at the graphic, "inviting" the student to do the same.

Another mixed-language explanatory technique could be used when the tutoring topic calls for the student to list a number of items, for example, possible confounds for an experiment. Using simple HTML commands, the tutor could jot the items down in a "notepad" area of the screen as the students names the items in order to reduce the student's working memory load.


Statistical Validity

Statistical Validity - experimental results are NOT due to chance, the changes in the DV are attributable to changes in the IV.
30
10
DV
Gap is a consequence of manipulation of the IV.
variance due to chance in Random Assignment
variance due to chance in Random Assignment
Low
High
IV

Figure 4. A graphic for explaining statistical validity

To evaluate RMT's graphical elements (including the talking head), we have started to use an eyetracker to analyze where and for how long a student's gaze is directed. We hope to correlate this data with eyetracker-based measures of the student's cognitive load [6] to identify aspects of the interaction that may be confusing to the student.

In this section we have presented a few ideas and a lot of questions. The use of mixed-language explanations has the potential to be extremely beneficial to students by helping them process and integrate new information, but significant research must be done to understand how it can best be deployed.

References

[1] B. S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13:4–16, 1984.

[2] R. Clark and R. Mayer. *e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning*. Pfeiffer, 2002.

[3] A.C. Graesser, G.T. Jackson, E.C. Mathews, H.H. Mitchell, A. Olney, M. Ventura, P. Chipman, D. Franceschetti, X. Hu, M.M. Louwerse, N.K. Person, and TRG. Why/autotutor: A test of learning gains from a physics tutor with natural language dialog. In *Proceedings of the 25th Annual Conference of the Cognitive Science Society*, Mahwah, NJ, 2003. Erlbaum.

[4] T.K. Landauer and S.T. Dumais. A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240, 1997.

[5] N. K. Person. *An analysis of the examples that tutors generate during naturalistic one-to-one tutoring sessions*. PhD thesis, University of Memphis, Memphis, TN, 1994.

[6] D. Salvucci and J. R. Anderson. Tracing eye movement protocols with cognitive process models. In *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*, pages 923–928, Hillsdale, NJ, 1998. Lawrence Erlbaum Associates.

[7] P. Wiemer-Hastings, K. Wiemer-Hastings, and A. Graesser. Improving an intelligent tutor's comprehension of students with Latent Semantic Analysis. In S. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education*, pages 535–542, Amsterdam, 1999. IOS Press.

Interpretation of mixed language input in a mathematics tutoring system

Helmut Horacek [a] Magdalena Wolska [b]
[a] *Fachrichtung Informatik*
[b] *Fachrichtung Computerlinguistik*
Universität des Saalandes, Postfach 15 11 50, D-66041 Saarbrücken, Germany

Abstract. Making effective use of communicating by natural language and formal expressions is demonstrated in a number of text books in formal domains. For tutorial systems, using such communication means would also be desirable, but this is substantially hindered by the inherent difficulties associated with interpreting these mixed language forms. In order to improve communication in tutorial systems, we have examined a corpus on teaching mathematical theorem proving in elementary set theory, in terms of functionalities required for handling the phenomena observed. Techniques required include an increased interactions between analysis components, extra knowledge needed for communication, and specific interpretations for imprecise terms and referring expressions. Although our case study only concerns a rather restricted domain, we are convinced that the findings carry over to other subareas of mathematics and related domains, such as naive physics.

1. Introduction

Text books in formal domains, such as mathematics, are characterized by a mixture of natural language and formal expressions. Communicating in this mixed language form turns out to be quite effective, combining the preciseness of formal expressions with occasional sloppiness in natural language descriptions. For tutorial systems, a prevailing trend has been to make the communication with the automated tutors as close to that of human–human tutoring as this has been proved to be the most effective in supporting the student's conscious learning process [4]. Hence, in formal domains, such communication means would also be desirable, but making use of them is substantially hindered by the inherent difficulties associated with interpreting natural language, not to mention further complications arising from the mixed language forms.

In this paper, we report on issues involved in interpreting mixed symbolic and natural language student utterances in the context of a student-tutor discourse on mathematical proofs and present strategies for the interpretation process. We examine a corpus on teaching proving skills in elementary set theory, in terms of functionalities required for handling the observed phenomena. The required techniques include an increased interactions between analysis components, extra knowledge needed for communication, and specific interpretations for imprecise terms and referring expressions.

The paper is organized as follows. First, we characterize the discourse phenomena. In the main part of the paper, we describe interpretation techniques for analyzing them, in particular, the role of external knowledge sources. Finally, we discuss future activities.

2. The language of mathematical proofs

In contrast to text books in formal areas such as mathematics, there is hardly any material on *dialogs* in such domains. To investigate the use of language in written dialogs on mathematical proofs, we have conducted an experiment with a simulated tutoring system for teaching proofs in naive set theory. In this section, we briefly present the setup of our corpus collection experiment and discuss the language phenomena.

2.1. Corpus collection

In a Wizard-Of-Oz (WOz) setup, we have asked 24 subjects, with varying educational background and little to fair mathematical knowledge, to solve proofs in naive set theory. The subjects were asked to prove three theorems: 1. $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$, 2. $A \cap B \in P((A \cup C) \cap (B \cup C))$, and 3. $If\ A \subseteq K(B),\ then\ B \subseteq K(A)$; K stands for set complement and P for power set. To encourage dialog with the system, they were instructed to enter proof steps, rather than complete proofs at once. Neither the subjects nor the tutor were restricted in the linguistic expression in formulating their turns. The dialogs were conducted in German (cf. [9] for details on the corpus).

2.2. Language phenomena

To illustrate the characteristics of the language used in our setting, we present an overview of the prominent language phenomena observed in our corpus. Examples are given in the original German version, accompanied with English glosses.

Semi-formal expressions One of the most prominent characteristics of informal proofs in our setting is that (semi-formal) mathematical expressions and natural language are tightly intermixed, *e.g.* (1), (3). However, utterances may also be worded entirely in natural language, (2). In a sloppy notation, mathematical expressions (or parts thereof) may lie within a scope of quantifiers or negation expressed in natural language, as in (4).

(1) $K(A \cup B)$ ist laut DeMorgan-1 $K(A) \cap K(B)$
According to DeMorgan-1 $K(A \cup B)$ equals $K(A) \cap K(B)$

(2) A enthaelt keinesfalls Elemente, die auch in B sind.
A contains no elements that are also in B

(3) $A \cap B$ ist $\in$ von $C \cup (A \cap B)$
$A \cap B$ is $\in$ of $C \cup (A \cap B)$

(4) B enthaelt kein $x \in A$
B contains no $x \in A$

Ambiguity Imprecise and informal descriptions formulated by students may frequently be associated with ambiguity. In (5), a structural ambiguity is introduced by an ambiguous coordination.[1] In (4), above, and (6), a multiply-ambiguous relation of *Containment* is introduced by predicates "contain" and "be in": on the one hand, possibly referring to STRUCTURAL COMPOSITION, on the other, to INCLUSION (in naive set theory still ambiguous between the relations of super-/subset or element), with both readings possible

[1]The alternative readings are: "[$x \in B$ and therefore $x \subseteq K(B)$] and [$x \subseteq K(A)$ given the assumption]" and "[[$x \in B$] and therefore [$x \subseteq K(B)$ and $x \subseteq K(A)$] [given the assumption]]"

in case of (4).[2] Finally, (7) and (8) exemplify ambiguities at the proof structure level. Both were uttered as first dialog turns on the part of the student in response to the first problem (see 2.1). In the former case, it is not clear whether the provided formula is to be interpreted as an instantiation of the DeMorgan rule or as a consequent of the formula to be proved. In (8), the point of application of the DeMorgan rule is not indicated.

(5) $x \in B$ *und somit* $x \subseteq K(B)$ *und* $x \subseteq K(A)$ *wegen Voraussetzung*
$x \in B$ and therefore $x \subseteq K(B)$ and $x \subseteq K(A)$ given the assumption

(6) *wenn* A *vereinigt* C *ein Durchschnitt von* B *vereinigt* C *ist, dann müssen alle* A *und* B *in* C *sein*
If the union of A and C is equal to intersection of B union C, then all A and B must be in C

(7) **S1:** *nach deMorgan-Regel-2 ist* $K((A \cup B) \cap (C \cup D)) = (K(A \cup B) \cup K(C \cup D))$
by deMorgan-Rule-2 it holds that …

(8) **S1:** deMorgan-Regel-2
deMorgan-Rule-2

Incompleteness In domains with formal expressions, references frequently exploit implicit metonymic relations, several examples of which are also found in our corpus. For instance, “left side”, (9), refers to a part of an equation that is not explicitly mentioned in the utterance. The expression “inner parenthesis”, (10), requires a metonymic interpretation as referring to the expression enclosed by that pair of parentheses. Similarly, the term “complement”, (11), does not refer to the operator per se, but to an expression identifiable by this operator, that is, where “complement” is the top-level operator. A type clash triggers the interpretation involving a metonymic extension. In (12), the expressions “smaller” and “larger” in relation to sets refer to the sets’ cardinalities rather than the sets themselves. Another form of incomplete specification is the use of intensifiers, which may require pragmatically-motivated interpretations. For example, the use of “entirely”, in (13), is intended to percolate the difference from the sets themselves to all their elements, as also indicated by the following explanatory remark.

(9) *Dann gilt für die linke Seite, wenn* $C \cup (A \cap B) = (A \cup C) \cap (B \cup C)$, *der Begriff* $A \cap B$ *dann ja schon dadrin und ist somit auch Element davon*

Then for the left hand side it holds that..., the term $A \cap B$ is already there, and so an element of it

(10) *Distributivität von Vereinigung über Durchschnitt:* $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ *Hier dann also:* $C \cup (A \cap B) = (A \cup C) \cap (B \cup C)$ *Dies für die innere Klammer*

Distributivity of union over intersection: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ Here: $C \cup (A \cap B) = (A \cup C) \cap (B \cup C)$ This for the inner parenthesis

(11) $K((A \cup B) \cap (C \cup D)) = K(A \cup B) \cup (C \cup D)$ *de morgan regel 2 auf beide komplemente angewendet*

$K((A \cup B) \cap (C \cup D)) = K(A \cup B) \cup (C \cup D)$ de morgan rule 2 applied to both complements

(12) *Der Schnitt von zwei Mengen ist kleiner gleich der kleineren dieser Mengen, also ist das Komplement des Schnitts größer gleich das Komplement der kleineren Menge*

The intersection of two sets is less or equal to the smaller of these sets, hence the complement of the intersection is larger or equal to the complement of the smaller set

(13) *dann sind* A *und* B *vollkommen verschieden, haben keine gemeinsamen Elemente*
then A and B are entirely different, have no common elements

Finally, there are also errors, specifically in formulas, which require dedicated treatment – for example: If $A \subseteq K(B)$ then $A \notin B$. For details about this issue cf. [1].

[2]If in the previous context there is an assignment of B to a formula in which $x \in A$ is a sub-expression, the STRUCTURAL COMPOSITION reading may be intended. The INCLUSION reading might be possible if B is a set whose elements are formulas.

3. Input interpretation

Our goal is to design an input interpretation strategy for a natural language discourse on mathematical proofs. In this section, we present the basic architecture and indicate how, it must be extended to support analysis of an increasing number of phenomena.

3.1. Baseline processing strategy

The basic processing pipeline involves four processing stages: (i) mathematical expression identification and parsing, (ii) syntactic and semantic sentence parsing (for more details cf. [8]), (iii) step-wise domain interpretation, (iv) formal representation.

Formula parsing The formula parser uses knowledge on domain relevant operation and identifier symbols. Identification of formula within natural language text is based on: single character tokens (including parenthesis), multiple-character tokens consisting only of relevant characters, mathematical symbol unicodes, and new-line characters.

The parser converts the infix notation used in the input into an expression tree based on which dedicated functions retrieve information on surface sub-structure (e.g., "left side" of an expression, list of sub-expressions, list of bracketed sub-expressions) and type (given the top level operator; e.g., CONSTANT-SYMBOL, VARIABLE-SYMBOL, TERM, FORMULA, 0_FORMULA (formula missing its left argument), etc.).

Syntactic and semantic analysis Linguistic analysis is conducted with OpenCCG[3], a parser for lexically-specified Combinatory Categorial Grammars, capable of compositional construction of linguistic meaning representations. Categorial Grammars are particularly useful for our purposes since they support analysis of coordinated structures. Moreover, mathematical expressions lend themselves to a categorial treatment in that, depending on their type, they occur in specific syntactic contexts. In particular, incomplete mathematical expressions may be represented as functional categories. For example, a formula missing a left argument (0_FORMULA, see above), is of category S\NP, that is: of a sentence expecting an NP category in the left context.

To account for mathematical expressions embedded within the natural language, the parser's lexicon encodes "generic" lexical entries for each mathematical expression type, together with plausible syntactic categories. The choice of syntactic categories was guided by a systematic study of syntactic contexts in which mathematical expressions are used, based on analysis of the corpus as well as course book mathematical texts. For example, the syntactic categories for a lexical entry FORMULA (corresponding to mathematical expressions of type FORMULA) are S, NP, and N.

The output of the parser is a relational structure representing a dependency-based deep semantics of the utterance. In our implementation, we use the Praguian set of dependency relations of the Functional Generative Description's tectogrammatical level [6].

Domain interpretation Domain interpretation of the of the linguistic meaning of the utterances is obtained using a SEMANTIC LEXICON, a collection of lexical semantic mapping rules that associate tectogrammatical frames with general concepts by indicating *functor to role-filler* mapping. Domain interpretation is subsequently obtained by interpreting the general concepts within the given task and the domain using a DOMAIN ON-

[3]http://openccg.sourceforge.net

TOLOGY. An example of a lexical rule is one that specifies that the *Norm* dependent, expresses a $Justification$ for a proof-step. Another example is a procedural interpretation of the word **gemeinsam** ("common"); see Figure 1.

3.2. Context-specific processing requirements

The basic analysis architecture only covers the simplest cases of mathematical discourse in our context (*e.g.* (1), (5), (13)). Hence, extensions are required, regarding:

parsing issues: incomplete mathematical expressions used as short-hand for natural language (3), use of spoken-language syntax while verbalizing mathematical expressions (6), scope phenomena involving parts of mathematical expressions (4);

lexical semantics and domain modeling: verbalizations (2), informal natural language formulations (12), imprecision and ambiguities introduced by natural language (6);

domain-specific reference resolution: modeling typographical properties of mathematical expressions (9), identifying referent candidates within mathematical expressions (11), domain-specific metonymic references (10), (11);

pragmatic disambiguation: relevance of domain reasoning in interpreting linguistically non-ambiguous, but contextually ambiguous proof step specifications (7), (8).

3.2.1. Parsing

Incomplete mathematical expressions Both the formula and the natural language parsers are adapted to support incomplete and/or malformed mathematical expressions and their interactions with the surrounding natural language text. In particular, the formula tagger and parser recovers information about incomplete expressions, using the knowledge on syntax and semantics of formal expressions in a given domain. For example, the operator $\in$, in (3), is tagged and, based on domain knowledge, identified to require two arguments where one is of type *inhabitant* and the other *set*. Accordingly, it is assigned a symbolic type 0_FORMULA_0 to indicate the missing left and right arguments. Furthermore, a corresponding lexical entry 0_FORMULA_0 of category S/PP_LEX:VON\N is included in the lexicon of the parser (sentence missing a *pp* to the right and an *n* to the left).

Interactions with natural language To account for interactions between mathematical expressions and the surrounding natural language text, as in (4), we identify structural parts of mathematical expressions that may be identified as part of or lie within the scope of a natural language expression. For mathematical expressions of type FORMULA, we split the expression at the top node (obtaining two alternative readings: "TERM$_1$ O_FORMULA$_1$" and "FORMULA_0$_2$ TERM$_2$") and re-interpret the resulting expressions in the context of the surrounding natural language text. The lexical entry for 0_FORMULA is of syntactic category S\N (and its semantics is SUCH THAT TERM HAS PROPERTY FORMULA), while the entry for FORMULA_0 is of category S/N. This re-interpretation of a mathematical expression allows us to obtain the intended reading of (4) (cf. [8]).

Domain-specific language syntax With (6), we illustrated the use of domain-specific syntax wording a formal expression in natural language. **Vereinigt** (Past Participle form of *unify*) is normally used with a prepositional phrase (**vereinigen mit +Dat.**), however, the presented construction is commonly used when formulas are

(1) $MAP((\textbf{contain}_{pred}, \mathrm{x}_{act}, \mathrm{y}_{pat}), (\text{CONTAINMENT}_{pred}, \text{container}_{act}, \text{contents}_{pat}))$

(2) $MAP((\textbf{contain}_{pred}, \mathrm{x}_{act:formula}, \mathrm{y}_{pat:formula}),$
$(\text{STRUCTURAL COMPOSITION}_{pred}, \text{structured object}_{act}, \text{substructure}_{pat}))$

(3) $MAP((\textbf{common}, \mathrm{p}_{pred,sem==\mathbf{have}}, \mathrm{x}_{act:coord(x_1,x_2,\ldots,x_n)}, \mathrm{y}_{pat:Pred}),$
$(\mathrm{Pred}(\mathrm{x}_1,\mathrm{y}_1) \wedge \mathrm{Pred}(\mathrm{x}_2, \mathrm{y}_1) \wedge \ldots \wedge \mathrm{Pred}(\mathrm{x}_n, \mathrm{y}_1)))$

(4) $MAP((\textbf{common}, \mathrm{p}_{pred,sem:Pred}, \mathrm{x}_{act:coord(x_1,x_2,\ldots,x_n)}, \mathrm{y}_{pat}),$
$(\mathrm{Pred}(\mathrm{x}_1, \mathrm{y}) \wedge \mathrm{Pred}(\mathrm{x}_2, \mathrm{y}) \wedge \ldots \wedge \mathrm{Pred}(\mathrm{x}_n, \mathrm{y})))$

(5) $MAP((\textbf{common}, \mathrm{p}_{pred,sem:Pred1}, \mathrm{x}_{act:coord(x_1,x_2,\ldots,x_n)}, \mathrm{y}_{pat:Pred2}),$
$(\mathrm{Pred1}(\mathrm{x}_1, \mathrm{y}_1) \wedge \mathrm{Pred1}(\mathrm{x}_2, \mathrm{y}_1) \wedge \ldots \wedge \mathrm{Pred1}(\mathrm{x}_n, \mathrm{y}_1) \wedge$
$\mathrm{Pred2}(\mathrm{x}_1, \mathrm{y}_1) \wedge \mathrm{Pred2}(\mathrm{x}_2, \mathrm{y}_1) \wedge \ldots \wedge \mathrm{Pred2}(\mathrm{x}_n, \mathrm{y}_1)))$

Figure 1. Example entries from the semantic lexicon

verbalized. To account for domain-specific constructions, we introduce domain-specific lexica in the parser's lexicon: the lexical entry for **vereinigt** first translates the item into its domain interpretation: 0_TERM_0 (similarly to the treatment of $\in$) with the syntactic category $\mathrm{S}\backslash\mathrm{NP}/\mathrm{NP}$.

3.2.2. Domain-specific anaphora

To resolve references to parts of mathematical expressions, first, the formula parser includes functions to recover parts of mathematical expressions in specific PART-OF relations to the original expression. The selection of identified formula parts is motivated by systematic reference in natural language to those parts and notably includes: typographical features, such as "sides" of terms and formula, linear orders (*e.g.* "first", "second" argument), structural groupings (bracketed sub-expressions). Second, the entities are represented in an ontological representation of the domain objects (cf. [2]).

3.2.3. Domain modeling

The ontological domain model is extended to include representation of imprecise and/or ambiguous concepts evoked by natural language descriptions as generalizations of mathematical relations in the considered domain. To mediate between ambiguous linguistic realizations of domain concepts, we use a linguistically-motivated SEMANTIC LEXICON that provides a mapping from dependency frames to domain-independent conceptual frames. The input structures are described in terms of tectogrammatical valency frames or specifications of relative heads/dependents of concept-evoking lexical items. The output structures are the evoked concepts with roles indexed by tectogrammatical frame elements or interpretation scripts. Where relevant, sortal information for role fillers is given. Figure 1 shows some lexicon entries explained below.

Containment The CONTAINMENT relation in its most common domain interpretation specializes into relations of (strict) SUBSET or ELEMENT. (1) It also has a specific interpretation in the context of STRUCTURED OBJECTS (such as FORMULA), where it expresses a relation of STRUCTURAL COMPOSITION between parts of the structure (SUBSTRUCTURE vs. the embedding STRUCTURED OBJECT). (2) It can be linguistically realized, among others, with the verb **enthalten** (*contain*). The tectogrammatical frame of **enthalten** involves the relations of *Actor* (*act*) and *Patient* (*pat*) that fill the CONTAINER and the CONTENTS roles respectively in the first reading, while in the second reading, *Patient* fills the SUBSTRUCTURE role.

Common property The semantics is derived using interpretation scripts based on the evoking lexical item **gemeinsam** (*common*). In Figure 1, Pred is a meta-object that can be instantiated with a relational noun (3), or a relational predicate (4). (5) is the most general case. Interpretation of (13) is instantiated with the first reading: $MAP((\textbf{common}, \mathrm{p}_{pred,sem==\textbf{have}}, \mathrm{x}_{act:coord(A,B)}, \mathrm{y}_{pat:Element}), (\textsc{element}(\mathrm{A},\mathrm{y}_1) \wedge \textsc{element}(\mathrm{B}, \mathrm{y}_1)))$

In order to interpret student utterances and to meet communication purposes, we manually constructed a DOMAIN ONTOLOGY, an intermediate representation that links concepts from the semantic lexicon with logical definitions represented in the domain knowledge base. The design and the role of the ontology is motivated in more detail in [2]. The core extensions include: 1. representation of *imprecise and general concepts* that need to be interpreted in domain-specific terms; 2. *typographical properties* of mathematical objects, including substructure delimiters (e.g. parentheses) and linear orderings, e.g. argument positions. The most interesting feature of the intermediate representation is incorporation of ambiguous and general concepts, such as CONTAINMENT, as SEMANTIC RELATIONS. For example, a CONTAINMENT holds between two entities if one includes the other as a whole, or all its components separately. This is a generalization of SUBSET and ELEMENT relations in naive set theory (see examples (4) and (6) in section 2.2). Moreover, as part of domain model, we encode polysemy rules for treatment of metonymic references. Examples of such rules in the mathematics domain are: FORMULA-SIDE : TERM-AT-SIDE, as in (9), BRACKET : BRACKETED-TERM, as in (10), OPERATOR : TERM-UNDER-OPERATOR, as in (11).

3.2.4. Disambiguation

In addition to the ambiguities discussed above, a proof-statement may yield different meanings according to its contextual interpretation. This interpretation comprises the relevance of the utterance as a contribution to the task goal and the likelihood that the most sensible interpretation is (1) indeed intended, (2) consciously specified by the student. Concerning the relevance of a proof-statement, we are able to test correctness and also coherence in terms of a partial solution built so far in the tutorial session. This is done by building and maintaining a representation of a proof constructed by the student.

A student may start to construct one proof, but in the course of the session, abandon it and attempt another proof, in which case, the system keeps track of the alternative solutions. It evaluates the appropriateness of the student's contributions with respect to a valid proof by attempting to incorporate the alternative interpretations of ambiguous statements into the proof state representation. For each alternative, the evaluation is one of the following categories: *relevant* (the contributed proof step is correct and brings the proof forward), (only) *correct* (statement true, but does not bring the proof forward), and *incorrect* (the contributed step is false). The current and preceding evaluations, provided the proof-steps relate to the same problem solution, direct a hinting algorithm for choosing an adequate system response in a socratic tutoring style [7].

In addition, the decision on which interpretation is intended is pragmatically influenced by other factors, such as: the student's knowledge of the domain concepts and their correct use, correct usage of domain terminology (*student model*) or contextual preference for one reading over the others. So far, we do not evaluate these factors. In the most "accommodating" approach, ambiguous statements evaluated as *correct* in one of the readings could be accepted without clarification, making dialog progression smoother.

4. Discussion and Conclusion

In this paper, we discussed phenomena related to communication by natural language and mathematical formulas in elementary set theory and elaborated on functionalities required to handle these phenomena. Several systems addressing subfields of formal domains, *e.g.* geometry (PACT [5]), electrical engineering (BEETLE [10]), and physics (Why2-Atlas [3]), also offer natural language based interaction, all with some degree of interaction with formal expressions. In comparison to these approaches, we can handle a tighter mixture of portions of natural language and parts of formal expressions. Moreover, the strict separation of knowledge sources enables us to treat varying forms of incompleteness and sloppiness in accordance with chosen tutorial strategies.

As dialogs in our corpus demonstrate, functionalities described in this paper are not sufficient to analyze and understand several of the students' statements which a human tutor would be able to do without much effort. Among the most severe problems are recognition of formula resp. natural language parts, which may sometimes be very tricky, determination of sometimes missing sentence boundaries, and dealing with recoverable errors in specifications. We intend to address these issues in the near future.

References

[1] H. Horacek and M. Wolska. Fault-tolerant interpretation of mathematical formulas in context. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED-05), poster*, Amsterdam, the Netherlands, 2005. To Appear.

[2] H. Horacek and M. Wolska. Interpreting semi-formal utterances in dialogs about mathematical proofs. *Data and Knowledge Engineering Journal*, 2005. To Appear.

[3] N. Makatchev, P. Jordan, and K. VanLehn. Modeling Students' Reasoning about Qualitative Physics: Heuristics for Abductive Proof Search In *Proceedings of the International Conference on Intelligent Tutorial Systems (ITS-04)*, pp. 699-709, Springer, 2004.

[4] J. Moore. What makes human explanations effective? In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, pp. 131–136, Hillsdale, NJ, 1993.

[5] O. Popescu and K. Koedinger. Towards understanding geometry explanations. In *Building Dialogue Systems for Tutorial Applications, Papers from the 2000 AAAI Fall Symposium*, pp. 80–86, Menlo Park, California, 2000. AAAI Press.

[6] P. Sgall, E. Hajičová, and J. Panevová. *The meaning of the sentence in its semantic and pragmatic aspects*. Reidel Publishing Company, Dordrecht, The Netherlands, 1986.

[7] D. Tsovaltzi, H. Horacek, and A. Fiedler. Building hint specifications in a NL tutorial system for mathematics. In *Proceedings of the 16th International Florida AI Research Society Conference (FLAIRS-04)*, pp. 929–934, Florida, USA, 2004.

[8] M. Wolska and I. Kruijff-Korbayová. Analysis of Mixed Natural and Symbolic Language Input in Mathematical Dialogs. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL-04)*, pp. 25–32, Barcelona, Spain, 2004.

[9] M. Wolska, B. Q. Vo, D. Tsovaltzi, I. Kruijff-Korbayová, E. Karagjosova, H. Horacek, M. Gabsdil, A. Fiedler, and C. Benzmüller. An annotated corpus of tutorial dialogs on mathematical theorem proving. In *Proceedings of International Conference on Language Resources and Evaluation (LREC-04)*, pp. 1007–1010, Lisbon, Portugal, 2004. ELDA.

[10] C. Zinn, J. Moore, M. Core, S. Varges, and K. Porayska-Pomsta. The BE&E Tutorial Learning Environment (BEETLE). In *Proceedings of the Seventh Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck-03)*, pp. 209–210, Saarbrücken, Germany, 2003.

Mixed Language Processing in the Why2-Atlas Tutoring System

Maxim Makatchev [1], Brian S. Hall, Pamela W. Jordan, Umarani Pappuswamy and Kurt VanLehn

Learning Research and Development Center, University of Pittsburgh

Abstract. When dealing with a natural language interaction about a formal domain, a number of phenomena occur. They include interspersing natural language with formulas, various degrees of formality, and conveying the logical structure of an essay. Capturing these phenomena, to some extent, is necessary for providing relevant tutoring feedback. In this paper we discuss these phenomena, the extent to which we process each of them in the Why2-Atlas tutoring system and directions for future work.

Keywords. Mixed languages, Dialog-based intelligent tutoring systems

1. Introduction

The Why2-Atlas tutoring system [10] is designed to encourage student's self-explanation via a natural language (NL) dialog. After a problem from the domain of qualitative mechanics is presented to the student, she is asked to write an essay containing her answer and explanation (Figure 1). The essay is analyzed for its correctness and coverage [7] and based on the found errors or missing facts or justifications the system starts a dialog. After the points are discussed in the dialog the student is asked to update her essay and the cycle continues until the essay is considered satisfactory.

When is the essay good enough? There are two tutoring objectives that are aimed at by Why2-Atlas. First, the student should use appropriate language to describe domain concepts. Second, the student should develop an understanding of relationships between mechanical concepts represented by laws and formulas and apply these relationships to generate an admissible logical argument that leads to the correct answer. These requirements imply the following criteria for evaluating an essay:

- proper use of physics terms,
- correctness of statements,
- correctness of justifications, namely laws and relationships,
- a significant overlap of the logical structure of the essay with the ideal solution graph (a "proof").

[1] Correspondence to: Maxim Makatchev, LRDC, 3939 O'Hara Street, Pittsburgh, PA 15260, USA. Tel.: +1 412 624 7498; Fax: +1 412 624 7904; E-mail: maxim@pitt.edu.

Problem: A heavy clay ball and a light clay ball are released in a vacuum from the same height at the same time. Which reaches the ground first? Explain.
Explanation: Both balls will hit at the same time. The only force acting on them is gravity because nothing touches them. The net force, then, is equal to the gravitational force. They have the same acceleration, g, because gravitational force=mass*g and f=ma, despite having different masses and net forces. If they have the same acceleration and same initial velocity of 0, they have the same final velocity because acceleration=(final-initial velocity) elapsed time. If they have the same acceleration, final, and initial velocities, they have the same average velocity. They have the same displacement because average velocity=displacementtime. The balls will travel together until the reach the ground.

Figure 1. The statement of the problem and a verbatim student explanation.

All of these evaluation criteria, except for the last one, apply also to the analysis of some of the student turns during the dialog stage. Analyzing an essay or a dialog utterance with respect to these criteria requires dealing with the following language phenomena:

- symbolic expressions: e.g. “a is constant,” “f=ma”;
- numeric expressions: e.g. “acceleration is 9.8 m/s^2”;
- formally and informally described physics terms: e.g. “speeding up” versus “speed is increasing,” “move side by side” versus “have equal positions”;
- generic and instantiated versions of the physics laws and formulas: “force on the ball equals its mass times g” versus “force is mass times acceleration.”
- logical relations being expressed in natural language and omission of logical relations: “*provided* there is no air resistance...,” “*if it was true that* the balls fall at the same acceleration, ...”

Similar language phenomena have been reported in tutoring systems for other formal domains, e. g. mathematical dialogs [1,11] and qualitative process theory [4].

In this paper we describe the extent to which each of these language phenomena are dealt with in the Why2-Atlas tutoring system. In Section 2 we discuss the representation and processing that we use for each of the language phenomena outlined above. Section 3 is dedicated to our the techniques for analyzing the student’s essay and dialog turns with respect to the evaluation criteria we listed above. We summarize our current progress and outline possible directions for future work in Section 4.

2. Recognizing and representing mixed language expressions

2.1. Symbolic and numeric expressions

Some examples of the expressions that fall into this category are:

a. “acceleration is final velocity minus initial velocity over elapsed time”
b. “net force is mass times acceleration”
c. “9.8 m/s^2”
d. “a = 9.8 m/s^2”
e. “the equation <net force = m * a>”

Formulas can be expressed in natural language *(a)*, *(b)*, in algebraic form *(c)*, *(d)*, or in natural language mixed with algebraic symbols *(e)*. In all cases these expressions are treated as semantic units and are marked in the student input by an equation identifier. The equation identifier matches the student's text with the stored representations of commonly occurring correct and erroneous formulas.

One of the observed differences between explanations in the subdomain of qualitative physics and the domain of mathematics is that much of the reasoning consists of application of a small number (37) of physics principles (10 of which are vector relations between physical quantities, 12 are their derivatives, including qualitative relations, and the rest are rules of idealization, e.g. "Possible forces are either contact forces or the gravitational force"). This allows us to match directly against the representations relevant to the correct and most common buggy formulas corresponding to the physics principles, as opposed to compositional representations of formulas as presented in [1]. In all, twelve legitimate and seven buggy (including "unrecognizable") mathematical forms are identified. The identifier is implemented as a series of regular expressions applied to the student input after spelling correction, but before invocation of any of our language understanding modules. The resulting text, with formulas replaced by tags can be passed through a parser such as CARMEL [8] or MINIPAR/Rappel [5,3] since both parsers are robust enough to skip unknown words (CARMEL), or treat them as nouns (MINIPAR), without significant anticipated performance loss. The equation identifier is currently being tested as a part of the evaluation of the Why2-Atlas system.

Since fragments of algebraic expressions can be interspersed with text, as in *(e)*, some degree of flexibility is needed in defining an acceptable syntax for formulas. The use of angle brackets as an enclosure for algebraic forms is implicitly suggested by incorporating this convention into the tutoring materials presented to the student. These grouping characters, together with parentheses, are recognized but optional (and need not even be balanced). Since mathematical forms in prose would tend to omit them even if the result was mathematically ambiguous, as in *(a)*, the equation identifier must tolerate this type of ill-formedness for all modes of expression. Currently lower and upper case distinctions are also ignored, even though it means losing the distinction between "G" (universal gravitational constant) and "g" (acceleration due to gravity).

The equation identifier is not entirely forgiving of ill-formedness, however; it detects a small set of common buggy formulas: for example, "a = m / f" instead of the correct "a = f / m." The idea is to identify common algebraic errors, and errors of omission such as "velocity" instead of "average velocity." The final step is a catch-all pattern to identify strings like "x = y * z" in which one or more of the variables cannot be recognized, as a generic buggy equation or perhaps more accurately, "unrecognizable mathematical form."

Example *(c)* is identified as a synonym for "g" gravitational acceleration, by virtue of both its numeric value and its units – the expression "9.8" on its own would not be given such privileged status. Such a numeric expression can, of course, be embedded in an equation proper, as in *(d)*.

Once the correct or buggy formula is identified, it is represented as an atom in the first-order predicate logic-based knowledge representation, for example
`(math-form mf1 a-equals-f-over-m)` and
`(math-form mf2 buggy-a-equals-f-over-m-inverted)`,
where `mf1` and `mf2` are the atom identifiers used for cross-referencing.

Category	Example of natural language expression
relative position	"keys are behind (in front of, above, under, close, far from, etc.) man"
motion	"move slower," "slow down," "moves along a straight line"
dependency	"horizontal speed will not depend on the force"
direction	"the force is downward"
interaction	"the man pushes the pumpkin," "the gravity pulls the ball"

Table 1. Categories of informal physics expressions.

2.2. Formal and informal physics

As with many other domains, some terms from the domain of mechanics have a much less formal use in everyday language, for example "this will force the objects to move at constant speed." This problem can normally be handled by filtering of syntactic categories. Conversely, many mechanics phenomena may be described in an informal language, for example "speed up" instead of "accelerate," "push" instead of "apply a force," "the force is downward" instead of "the force is negative." This type of informality has proven to be a significant problem.

The difficulty is partially in defining the boundaries of equivalence classes for informal expressions about physics concepts. Consider the example "The object will slow down." The correctness of this statement with respect to a particular context can be evaluated, despite its somewhat informal language, by representing it as a velocity with a decreasing magnitude, a decreasing speed, or a negative acceleration. When evaluating the coverage of this statement, however, it may not be desirable to represent this statement in terms of formal physics, since this may attribute to the student more knowledge than she has actually expressed.

We address this problem by adopting representations for different levels of formality. The formal physics concepts are represented by predicates for vector quantities (position, displacement, velocity, acceleration, force, total force, momentum), scalar quantities (mass, speed, distance, duration), states (for example contact state, being in vacuum, freefall), relations (comparison of magnitudes and directions) and a predicate defining the order of the time instants [6]. Informal expressions can be grouped into categories, some of which are shown in Table 1.

In the current version of our system we use a dedicated predicate for representing each of these informal categories, except for the interaction category which we are considering implementing at a later stage. Below we include the somewhat abridged representation of the sentences "there is a downward force of gravity"

```
 (force f1 ?body1 ?body2 ?comp1 ?d-mag1 ?d-mag-num1
       ?mag-zero1 ?mag-num1 ?dir1 ?dir-num1 ?d-dir1 ?time1 ?time2)
(due-to dt1 f1 gravity)
(coordinate-system cs1 ?dir1 down)
```

and "the keys are behind the man"

```
 (rel-position rp1 behind keys man ?time3 ?time4)
```

In the former example arguments that are equal across predicates are represented via shared variables.

2.3. Generic and instantiated physical laws

The qualitative rules of physics can be described in a generic form, e.g. “for two objects, if the acceleration, initial velocity and duration are the same, so is final velocity,” or in an instantiated form “the balls have the same initial velocity and acceleration so their velocity will be the same at all times.” Formulas (both in algebraic and NL form) can also be generic “a=(vf-vi)/t” or specific (for example, “a1=vf1-vi1/t1,” “acceleration of the heavy ball equals to final velocity minus initial velocity over t.”)

Generic formulas are recognized and represented using the framework described in Section 2.1. While we have not seen many specific algebraic expressions in our corpus, there are cases when formulas expressed in natural language refer to specific bodies. This imbalance can be explained by two facts: first, there are no examples of specific algebraic expressions used by the tutor; second, specific algebraic expressions require introducing new notations, while specific formulas expressed in NL do not. Due to the small number of observed examples in our corpus, currently we do not attempt to represent or recognize instantiated formulas in either algebraic or NL form.

However we do represent to a certain degree both generic and specific versions of the qualitative relations that are consequences of the formulas. Consider the examples of a generic and its respective specific relation from the beginning of the section: “For two objects, if the acceleration, initial velocity and duration are the same, so is final velocity,” and ‘the balls have the same initial velocity and acceleration so their velocity will be the same at all times.” The difference between these two expressions is in the specificity of the subject of the sentence: a ball is a subclass of an object. This nuance can be represented by appropriate type restrictions on the corresponding variables and constants, and by using a universal quantifier. Our current knowledge representation, while allowing for typing of variables, does not allow quantifiers or Skolem constants, for the sake of efficiency of reasoning. Instead, all the variables in standalone atoms are assumed to be existentially quantified and all variables in the rules (except for variables present only in the consequent of a rule) are assumed to be universally quantified [6]. Therefore currently we settle for an imprecise representation of the generic propositions, via an existentially quantified variable of a generic type. While this would lead to ambiguity, in that an existentially quantified version of the statement would have a representation that is identical with that of the universally quantified version, lack of the existentially quantified expressions in our corpus (of the sort “there are two objects that have the same velocity”) alleviates the potential problem. Incorporating a syntax for both quantifiers is planned for a future version of the knowledge representation.

2.4. Recovering the logical structure

Ideally, we would like to treat an essay not as a set of unordered propositions about the domain, but to capture and represent the causal and dependency relations between these propositions. Representing causal relations is complicated by the fact that there are potentially multiple causes for a particular consequence. Aside from the difficulty of recognizing the causes from the text, there is the problem of representing the cause-effect relation in a form that (1) can be flexible enough to account for a variable number of causes, (2) can be efficiently processed. One solution that satisfies both of these criteria would require a predicate of variable arity:

`(cause c1 cause1 cause2 cause3 ...causeN effect1)`.
This representation is considered superior to one with N binary predicates of the form
`(cause c1 causei effect1)`
that increases the number of cross-referenced atoms to be matched against stored representations. Matching cross-referenced atoms is an expensive procedure: the time complexity of the algorithm is $O(2^n n^3)$, where n is the number of input atoms [9].

Another nuance of cause-effect representation is *asserting* versus *non-asserting* conditions. Consider the following examples, "if there was air resistance, the larger ball would fall faster," and "since there is no air resistance, the balls fall at the same speed." Clearly there is a difference in the speaker's belief about whether the condition actually holds or not. The logical structures corresponding to these sentences can be represented as $A \rightarrow B$ and $A \wedge (A \rightarrow B)$ respectively.

Currently we do not recognize this type of logical structure in an essay, and instead match the unordered set of cross-referenced atoms with the stored representations for facts and physics rules and analyze the intersection of the matched facts and rules with the statements in the nodes of an ideal "proof" graph (more details are in Section 3.1).

One aspect of essay structure that we do represent is dependencies between physical quantities. This is relevant for representing such physics statements as "the freefall acceleration does not depend on mass," "the horizontal velocity does not depend on the vertical force," etc., and was discussed in Section 2.2. We are considering implementing a more sophisticated mechanism for reasoning about the logical structure of an essay in a future version of the system.

3. Evaluating essays and dialog turns

3.1. Coverage

Each of the four physics problems implemented in the system has an ideal "proof" designed by expert physics tutors that contains steps of reasoning, i.e. facts and their justifications, and ends with the correct answer. A fragment of the proof for the Clay Balls problem stated in Figure 1 is given in Figure 2. This proof is represented in the system by a graph such that an edge (a, b) is in the graph if a is a justification rule or a fact that is used as a premise (antecedent) of a rule that derives b. The justification in the ideal proof corresponds to the generic physics rules, in their algebraic form and/or qualitative form. The representation of a student essay as a set of cross-referenced first-order predicate logic atoms is matched against the nodes of the ideal graph, some of which are marked as required. Based on the overlap with the required nodes and taking into account the structure of the graph, the system initiates a dialog to elicit the remaining required points [10].

In addition to the manually generated ideal proof graph, each problem has a corresponding automatically generated assumption-based truth maintenance system (ATMS) that includes a large set of possible correct and buggy facts that can be derived from the problem givens using correct and buggy rules. This allows us to compute the inferential proximity metric between the student utterance and a required fact [7]. If the utterance is within a single inference step from a required fact the student may be given partial credit and appropriate feedback to help bridge the gap between the two statements. Providing this kind of a feedback is being considered for a future version of our system [2].

Step	Proposition	Justification
1	Both balls are near earth	Unless the problem says otherwise, assume objects are near earth
2	Both balls have a gravitational force on them due to the earth	If an object is near earth, it has a gravitational force on it due to the earth
3	There is no force due to air friction on the balls	When an object is in a vacuum, no air touches it
4	The only force on the balls is the force of gravity	Forces are either contact forces or the gravitational force
5	The net force on each ball equals the force of gravity on it	[net force = sum of forces], so if each object has only one force on it, then the object's net force equals the force on it
6	**Gravitational force is w = m*g for each ball**	**The force of gravity on an object has a magnitude of its mass times g, where g is the gravitational acceleration**
⋮	⋮	⋮
18	**The balls have the same initial vertical position**	given
19	The balls have the same vertical position at all times	[Displacement = difference in position], so if the initial positions of two objects are the same and their displacements are the same, then so is their final position
20	**The balls reach the ground at the same time**	

Figure 2. A fragment of an ideal "proof" for the Clay Balls problem from Figure 1. The required points are in bold.

3.2. Correctness

Domain statements that are recognized from the student input, namely facts (including instantiated rules) and generic rules, are also analyzed for correctness. This is done via a two step process: first, the statements are matched against known common buggy statements; second, they are matched against nodes of the ATMS. The latter match allows us to determine that a statement is buggy even when there is no corresponding bug in the list of common bugs, because the only ATMS environments in which it holds true include buggy assumptions. More details on this, including preliminary evaluation results, can be found in [7].

4. Conclusion and future work

Natural language interaction with a student about a formal domain produces a number of interesting natural language phenomena that need to be processed to generate adequate tutoring feedback. In our system we demonstrated the feasibility of capturing those phenomena that require limited knowledge about the domain and are undamaged by our treatment of an essay as a set of individual sentences. This already provides the system with useful information for generating richer tutoring feedback. Among the challenging phenomena that we hope to tackle in future versions of the system are the logical structure of an essay and more complete treatment of informal statements about the domain.

Acknowledgments

This research has been supported under NSF grant 0325054 and ONR grant N00014-00-1-0600. The authors would like to thank present and past members of the Natural Language Tutoring group for their work on the Why2-Atlas tutoring system.

References

[1] Helmut Horacek and Magdalena Wolska. Interpreting semi-formal utterances in dialogs about mathematical proofs. In F. Meziane and E. Métais, editors, *Natural Language Processing and Information Systems*, volume 3136 of *LNCS*, pages 26–38. Springer, 2004.

[2] Pamela W. Jordan. Using student explanations as models for adapting tutorial dialogues. In *Proceedings of 17th International FLAIRS Conference*, 2004.

[3] Pamela W. Jordan, Maxim Makatchev, and Kurt VanLehn. Combining competing language understanding approaches in an intelligent tutoring system. In *Proceedings of Intelligent Tutoring Systems Conference*, volume 3220 of *LNCS*, pages 346–357, Maceió, Alagoas, Brazil, 2004. Springer.

[4] Sven E. Kuehne and Kenneth D. Forbus. Capturing QP-relevant information from natural language text. In Johan de Kleer and Kenneth D. Forbus, editors, *Proceedings of the 18th International Workshop on Qualitative Reasoning*, pages 25–32, Evanston, USA, 2004. Lawrence Erlbaum Associates.

[5] Dekang Lin. Dependency-based evaluation of MINIPAR. In *Proc. of Workshop on the Evaluation of Parsing Systems*, Granada, Spain, May 1998.

[6] Maxim Makatchev, Pamela W. Jordan, and Kurt VanLehn. Abductive theorem proving for analyzing student explanations to guide feedback in intelligent tutoring systems. *Journal of Automated Reasoning, Special issue on Automated Reasoning and Theorem Proving in Education*, 32:187–226, 2004.

[7] Maxim Makatchev and Kurt VanLehn. Analyzing completeness and correctness of utterances using an ATMS. In *Proceedings of Int. Conference on Artificial Intelligence in Education, AIED2005*. IOS Press, July 2005.

[8] Carolyn P. Rosé. A framework for robust semantic interpretation. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 311–318, 2000.

[9] Kim Shearer, Horst Bunke, and Svetha Venkatesh. Video indexing and similarity retrieval by largest common subgraph detection using decision trees. *Pattern Recognition*, 34(5):1075–1091, 2001.

[10] Kurt VanLehn, Pamela Jordan, Carolyn Rosé, Dumisizwe Bhembe, Michael Böttner, Andy Gaydos, Maxim Makatchev, Umarani Pappuswamy, Michael Ringenberg, Antonio Roque, Stephanie Siler, and Ramesh Srivastava. The architecture of Why2-Atlas: A coach for qualitative physics essay writing. In *Proceedings of Intelligent Tutoring Systems Conference*, volume 2363 of *LNCS*, pages 158–167. Springer, 2002.

[11] Magdalena Wolska and Ivana Kruijff-Korbayová. Analysis of mixed natural and symbolic language input in mathematical dialogs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 25–32, Barcelona, Spain, 2004.

Relating Student Text to Ideal Proofs: Issues of Efficiency of Expression[1]

Pamela W. Jordan [2], Maxim Makatchev and Umarani Pappuswamy
University of Pittsburgh, Learning Research and Development Center
3939 O'Hara Street, Pittsburgh PA, 15260, USA.

Abstract. In this paper we focus on text analysis issues and describe the linguistic problems we have encountered in working with an ideal two-column proof as the pedagogical foundation for the tutoring system. We discuss how the student's explanation is challenging to assess relative to an ideal two-column proof since in the explanation (1) parts of the proof can be skipped and may not necessarily reflect a gap in the student's knowledge and (2) parts of the proof can be merged. We hypothesize that skipping and merging are due to efficiency of expression and begin to examine the implications for assessing the depth of a student's understanding of the domain. Finally we outline the solutions we have attempted and are considering for these issues.

Keywords. Text analysis, Mixed-language explanations

1. Introduction

One goal of the Why2 project has been to experiment with and evaluate the effectiveness of a variety of NLP techniques for tutoring. We selected qualitative physics as the tutoring domain as it required students to generate extended natural language explanations. To facilitate this experimentation we divided the interaction into two phases, (1) entry of an essay by a student that answers and justifies the answer to a qualitative physics problem and (2) a follow-up dialogue that over an extended interaction will help the student remedy flaws in his essay. In this paper we will focus on how the Why2-Atlas system addresses the first phase.

In the most recent version of Why2-Atlas we adopted a goal of eliciting an essay that covers a subset of an ideal two-column proof, where physics principles in the right column are used as justifications of the facts in the left column. Previous versions of the systems elicited a subset of the facts in the left column [11]. A hypothesized advantage of requiring discussion of justifications is that it encourages deeper learning because physics principles are now explicitly exercised. Although so far we have collected 48 problem interactions between students and the current version of the system, we have not yet started any formal studies of students' language use. Instead, we show some excerpts

[1]This work was funded by NSF grant 0325054 and ONR grant N00014-00-1-0600.
[2]Correspondence to: Pamela Jordan, Tel.: +1 412 624 7459; Fax: +1 412 624 7904; E-mail: pjordan@pitt.edu.

Question: Suppose a man is in an elevator that is falling without anything touching it (ignore the air, too). He holds his keys motionless right in front of his face and then lets go. He neither tosses them up nor throws them down; he just releases his grip on them. What will happen to them? Explain.

Prescribed Explanation: (18f) The keys remain in front of the man's face the whole way down. We can show this by analyzing forces and motions along the vertical dimension. (12f) Before the release of the keys the man and the keys have the same velocity because they are moving together. (4f) After the release, the only forces on the man and the keys are gravitational. (6f) Thus, their net forces are equal to their gravitational forces. (5jv) Now because <gravitational force = mass * g> and (8jv) <net force = mass * acceleration>, we know that (10f) <acceleration = g> for both the man and the keys. (13) Because their accelerations are the same, and their initial velocities are the same, the man and the keys have the same final velocity in accordance with <acceleration = (final velocity - initial velocity) /elapsed time>. (14) Because their acceleration is constant and they have the same initial and final velocity, we know that the man and the keys have the same average velocity as well. (15) This implies that they have the same displacements at all time, because <average velocity = displacement / elapsed time>. (16f)(17fjq) Because the man's face and the keys start at the same height, and they have the same displacement at all times, they have the same vertical position at all times. (18) Thus the keys remain in front of the man's face during the whole trip down."

Figure 1. The statement of the problem and its prescribed essay. The numbered codes are insertions we have added that relate a sentence or clause to parts of the ideal proof.

from the corpus to illustrate the linguistic problems we have encountered in working with the two column proof as the pedagogical foundation for the tutoring system. We will discuss how the student's explanation is challenging to assess relative to the ideal proof since in the explanation (1) parts of the proof can be skipped and this may not necessarily reflect a gap in the student's knowledge and (2) parts of the proof can be merged.

First we will give an overview of Why2-Atlas. Next we describe the issues involved in evaluating a student essay relative to the ideal two-column proof in which steps may have been omitted or merged due to efficiency of expression. Finally we outline the solutions we have attempted and future work we are considering.

2. The Why2-Atlas System

Why2-Atlas covers four qualitative physics problems on introductory mechanics. When the system presents one of these problems to a student, it asks that he type an answer and explanation and informs him it will analyze his final response and discuss it with him. One of the problems Why2-Atlas covers is shown in Figure 1 along with the prescribed ideal response[1], which the student is shown before moving on to a new physics problem. An initial verbatim student response is shown in Figure 2 and the response from the same student after several follow-up dialogues with Why2-Atlas is shown in Figure 3. Each of the essays shown has numbered codes inserted for exposition purposes that we will explain at the end of this section.

[1]The domain experts simplified some aspects of the explanation and made others more complicated in order to balance exposure to principles during evaluations of the system.

(18) They will'hang in the air'in front of him. (12f-14f) Both he and the keys are falling at the same velocity, And no force is being exerted on the keys by the man, (17f) so they'll be right next to one another as they fall.

Figure 2. An initial verbatim student explanation for the problem in Figure 1. The numbered codes are insertions we have added that relate a sentence or clause to parts of the ideal proof.

(4f) The only force acting on both man and keys is gravity. (5f) The magnitude of the force of gravity on the man and the keys is its mass times g. (7f) The magnitude of the net force on each body equals its mass times g.
(8jv) <net force = mass * acceleration>. (8f) Therefore, the magnitude of both accelerations is f/m. (10f) This is equal to g.
(13jv)<Acceleration = (final velocity - initial velocity)/elapsed time>. (12f) The initial velocity of the keys is the same as the initial velocity of the man. (13f) The final velocities are also the same. The time is the same. (13jq) If the acceleration, initial velocity, and time are the same, then the final velocity is too. (15jv) <Average velocity = displacement / elapsed time> (14f) The average velocities are also the same. (17jq) If two things have the same v and t, then they have the same d.

Figure 3. A verbatim subsequent explanation from the same student in Figure 2 for the problem in Figure 1. The numbered codes are insertions we have added that relate a sentence or clause to parts of the ideal proof.

During the essay analysis phase, each sentence entered by the student is first subjected to a pre-processor that segments complex sentences, marks up equations, such as <average velocity = displacement / elapsed time>, and corrects spelling errors. Each corrected, marked-up sentence segment is then competitively analyzed by three different sentence-segment analysis techniques and a final interpretation is heuristically selected [6]. The final output of the sentence segment analysis is a function-free first-order predicate logic (FOPL) representation for each sentence segment.

Our intent in applying multiple sentence-segment analysis approaches is to use each to its best advantage relative to a particular time-slice in the life-cycle of the knowledge development effort for the tutoring system. At a given time-slice one approach may be functioning better than another for certain types of sentence segments. But since the knowledge development is on-going, we anticipate that the performance may change over time.

The selection heuristics for choosing which result to use depend on the FOPL representation language but not on the analysis techniques. The heuristics filter and rank each output representation using an estimate of whether a resulting representation either over or under represents the segment. The estimate combines counts of matches between the root forms of the words in the natural language segment and the constants in the FOPL representation returned by each method.

While any number of analysis approaches could be incorporated into the system in this way, we are currently using three that represent a range of approaches: symbolic, statistical and a hybrid. The statistical approach uses classes defined relative to the two-column proof and thus the FOPL representation for each class is stored and accessed when needed by the selection heuristics. The other two approaches directly produce an FOPL representation.

As the final step in analyzing a student's essay, an assessment of correctness and completeness is performed by matching the final FOPL representations of the student's essay to nodes of an augmented assumption-based truth maintenance system (ATMS) [8].[2] An ATMS for each physics problem is generated off-line. Both good and buggy physics rules are applied to the givens specified in the problem statement. Each anticipated student misconception is treated as an assumption (in the ATMS sense), and all conclusions that follow from it are tagged with a label that includes this assumption along with any other assumptions needed to derive that conclusion.

Completeness in Why2-Atlas is relative to an ideal two-column proof generated by a domain expert. The ideal proof for the problem in Figure 1 is shown in Figure 4 where facts appear in the left column and justifications that are physics principles appear in the right column. Justifications are further categorized as vector equations (e.g. <Average velocity = displacement / elapsed time>, in step (15) of the proof), or qualitative rules (e.g. “so if average velocity and time are the same, so is displacement” in step (15)). A two-column proof is represented in the system as a directed graph in which nodes are facts, vector equations, or qualitative rules that have been translated to the FOPL representation language off-line. The edges of the graph represent the inference relations between the premise and conclusion of modus ponens.

Matches of input representations against the ATMS and the two-column proof do not have to be exact. Further flexibility in the matching process is provided by examining an inferential neighborhood of radius N (in terms of graph distance) from matched nodes in the ATMS to determine whether it contains any of the nodes of the two-column proof. This provides an estimate of the inferential proximity of a student’s utterance to nodes of the two-column proof. Details of the analysis of correctness and completeness of the essay are provided in [8] and will not be covered further in this paper.

To determine which nodes are minimally required to be covered in a student essay, initially, our pedagogical expert for the domain estimated that for the four problems addressed by the system, a minimally acceptable essay should include all the facts (left column) and a subset of the justifications (right column). In the case of the problem shown in Figure 1, the minimum required subset of justifications is the three justifications in bold in Figure 4. The decision to require just a subset of the justifications was motivated by the purely practical need to limit the time that a student would have to spend in the worse case on a particular problem (i.e. schedule and budget constraints). Thus for this selection process we used a rule of thumb to select only those justifications that involved the most fundamental physics principles. The intuition is that not all justifications are of equal importance in the learning process.

In the remainder of the paper, we will refer to an entire proof step using its step number but to refer to parts of a step we will append to the step number “f” for the fact, and “j” for the justification. To refer to the parts of a justification we will append after the “j”, “v” for the vector equation and “q” for the qualitative statement. For example, (15) refers the full step 15 in the proof, (15f) refers to just “The keys and the man have the same displacements at all times”, (15jv) refers to just “<Average velocity = displacement / elapsed time>”, (15jq) to just “so if average velocity and time are the same,so is displacement”, and (15j) to the combination of (15jv) and (15jq).

[2]This matching step is skipped for any sentence-segment result that is a product of the statistical approach.

Step	Fact	Justification
1	The keys and the man have a gravitational force on them due to earth	If an object is near earth, it has a gravitational force on it due to the earth
2	There is no force on the keys due to air friction	The force due to air resistance is zero when there is no relative motion between air and the object
3	There is no force on the man due to air friction	The force due to air resistance is zero when there is no relative motion between the air and the object
4	The only force on the keys and the man is the force of gravity	Forces are either contact forces or the gravitational force
5	The magnitude of the force of gravity on the man and the keys is its mass times g	The force of gravity on an object has a magnitude of its mass times g, where g is the gravitational acceleration
6	The net force on each body equals the force of gravity on it	¡net force = sum of forces¿, so if an object has only one force on it, then the object's net force equals the force on it
7	The magnitude of the net force on each body equals its mass times g	Transitivity: if A=B and B=C, then A=C; if A=B and B<C, then A<C, etc.
8	The magnitude of each body's acceleration equals its net force divided by its mass	**<net force = mass * acceleration>, so the magnitude of the net force on an object equals its mass times the magnitude of its acceleration**
9	The magnitude of each body's acceleration equals its mass * g divided by its mass	Transitivity: if A=B and B=C, then A=C; if A=B and B<C, then A<C, etc.
10	The magnitude of each body's acceleration equals g	Canceling out: If A=B*C/C then A=B
11	The key and the man have the same acceleration	Transitivity: if A=B and B=C, then A=C; if A=B and B<C, then A<C, etc.
12	Because the man was holding the keys initially, and he moves along with the elevator, the keys and the elevator have the same initial velocity	If an object moves along with an agent, they have the same velocity, acceleration and displacement
13	At every time interval, the keys and the man have the same final velocity	**<Acceleration = (final velocity - initial velocity)/elapsed time>, so for two objects, if the acceleration, initial velocity and time are the same, so is final velocity**.
14	The man and the keys have the same average velocity while falling	If acceleration is constant, then <average velocity = (vf+vi)/2>, so if two objects have the same vf and vi, then their average velocity is the same.
15	The keys and the man have the same displacements at all times	**<Average velocity = displacement / elapsed time>, so if average velocity and time are the same, so is displacement**.
16	The keys and the man have the same initial vertical position	given
17	The keys and the man have the same vertical position at all times	<Displacement = difference in position>, so if the initial positions of two objects are the same and their displacements are the same, then so is their final position
18	The keys stay in front of the man's face at all times	

Figure 4. The ideal "proof" used in Why2-Atlas for the Elevator problem in Figure 1. The prescriptively required justifications are in bold.

3. Issues in Relating Text to Proofs

Note that while the prescriptive and subsequent student essays do include both facts and justifications, relative to the ideal proof many facts are skipped, parts of justifications are skipped and parts of multiple steps are merged. Looking at the numbers that we have inserted in the prescribed essay, note that proof parts (1f)-(3f),(7f),(9f),(11f) and (16f) are missing from the essay. Note too that some of these facts are also missing from both of the student's essays. In (13)-(15) of the prescriptive essay, the qualitative part of a justification and the fact that it helps justify are merged so that only the specific fact is mentioned. In the case of (5jv),(8jv) and (10f) in the prescriptive essay, these step components are merged into one sentence.

We will start first with skipped facts. We claim that fact (1f) is skipped in both the prescriptive and student essays, because saying the fact (4f) pragmatically presupposes the fact (1f). Although there is no settled formal definition of a pragmatic presupposition, loosely, it is any background assumption that arises for a statement [7]. For example, a presupposition of (4f) is "there exists a force due to gravity on the keys" because the definite noun phrase is thought to act as a presuppositional trigger [7].

Furthermore, we claim that explicitly stating both (4f) and (1f) in natural language is in violation of Grice's Maxim of Quantity (say no more than is necessary) [4] and thus it should be unlikely for both to appear together in a written explanation. Grice hypothesized four Maxims of Conversation: (1) Quality; say only what you believe to be true and have evidence for, (2) Quantity; say just the right amount relative to the purpose of the exchange, (3) Relevance; be relevant, (4) Manner; be brief, orderly and avoid obscurity and ambiguity.[3] He also hypothesized that seeming violations of these maxims would lead a hearer to seek an interpretation that would resolve the violation, thus leading to conversational implicatures. We claim these maxims and responses to violations are relevant for text as well given that text can be viewed as a variant of conversation [2].

We see that students do clearly violate some of these maxims. For example, during dialogues with Why2-Atlas, we saw many cases of students violating the Maxim of Quality. When the system requested that they provide evidence for an answer they just gave, students sometimes admitted they guessed at the answer. But what, if any, useful conversational implicatures the tutor should make for seeming violations remain to be determined. In the case of a violation of the Maxim of Quantity, in which more than is necessary is said, it may be a helpful diagnostic of the student's grasp of the material. Including what is typically viewed as unnecessary redundancy could be interpreted as the student failing to see the obvious connections or failing to know the premises for a rule. So it may be best to treat what should be pragmatic presuppositions as optional and their inclusion in a text is perhaps indicative of a student who needs help in achieving a more coherent picture of the reasoning process.

Facts (2f) and (3f) are also typically left out of student essays because both are givens and explicitly indicate that a force is to be ignored. Because this is near the beginning of the problem, saying (4f) means it is likely the given information was still salient for the student and that if the student considered all types of forces then (2f) and (3f) were used in reasoning. Another given (16f), appears in the prescriptive essay as part of (17q). Here the problem statement is no longer as salient as at the beginning of the text. Finally, we

[3]Although the maxims provide insight into why particular content is made explicit or not, they offer little guidance for implementation.

note that (7f), (9f) and (11f) each are conclusions of simple math manipulations and thus their inclusion could be a signal that the student believes these are of equal or greater importance than the other steps. Work in natural language generation operationalizes related omissions of intermediate reasoning and parts of rules used during reasoning by appealing in part to a model of the user's knowledge and attention [5].

Looking next at merged parts of the proof, we claim that this is related to Grice's Maxim of Manner (brevity) and again appeal to work in natural language generation to explain why this efficiency of expression is a possible display of brevity. A goal-directed view of sentence generation suggests that speakers can attempt to satisfy multiple goals with each utterance [1] and thus the same form can opportunistically contribute to the satisfaction of multiple goals [10]. But there are trade-offs across linguistic levels so that an intention which is achieved by complicating a form at one level may allow the speaker to simplify another level by omitting important information. (E.g. a choice of clausal connectives at the pragmatic level can simplify the syntactic level [3]).

Within a language generation system overloading is accomplished in part during the *sentence aggregation* process [9]. Sentence aggregation optionally combines simple sentences into more complex ones and often improves coherency of a text. So we claim that aggregation may reflect coherency in the mind of the student and that short, choppy, non-aggregated sentences may indicate the student is having trouble grasping the material. However, the student may avoid aggregation if text analysis frequently fails to understand his aggregated sentences. Note that the initial student essay is more aggregated than the subsequent one and that the short sentences are very similar to the corresponding part of the two-column proof. The system often bottoms-out and explicitly tells the student what is was trying to get him to add to his essay.

We also see semantic aggregation as well, in that specific and generic content are frequently merged so that we see the more specific content more than just generic (as in (17jq) in Figure 3) or a combination of generic and specific (as with (5jv),(8jv) and (10f) in Figure 1).

4. Initial Solutions and Future Work

With respect to which parts of the proof can be skipped in students' explanations, all parts of the ideal two-column proof are represented within the system but just those parts that are minimally required are marked as required. Thus the student is permitted to skip any part of the proof that is not marked as required. The dialogue then only addresses the marked nodes of the proof that could not be linked to the student text. While deciding what to mark as required, we made judgments about when steps are presupposed and only marked facts that are not presuppositions or salient givens. We did still mark simple math conclusions such as (10f) as required but in retrospect perhaps should not have.

With respect to parts of the proof being merged in the student essay, the system must handle one-to-many mappings between a sentence and parts of a proof. We currently have no mechanism to distinguish acceptable and unacceptable aggregations and perform only an assessment of completeness by matching representations of what the student said to an ideal proof and the ATMS. A problem with this matching process arises if the selected sentence-segment analysis is a product of the statistical approach. Recall that the statistical approach classifies a sentence-segment according to classes that are the nodes of the proof graph representation. But the statistical approach currently returns just the single most likely class. Thus when sentence segmentation fails and steps are merged in the sentence, part of the content of the sentence will be missed and will not be matched to the proof. This is not a problem for the other two sentence-segmentation approaches. However, currently the output of the statistical approach is the result that is most frequently selected by the heuristics. Thus it is worthwhile investigating whether we can automatically recognize contexts in which this approach should return multiple classifications in order to see if it will improve the accuracy of linking sentences to the two-column proof.

Further, while the sentence-segmenter may break sentences into clauses appropriate to parts of the proof representation, it was not specifically developed to support segmentation particular to the two-column proofs. It was instead developed to improve parsing efficiency by breaking complex sentences into simpler ones. We are considering retraining the sentence-segmenter relative to the parts of the proof as this should still accomplish its original purpose and potentially improve the ability to handle one-to-many mappings.

Finally, we will consider the extent to which proof transformation methods used in NL generation [5] can be effectively reversed for analysis purposes.

References

[1] Douglas E. Appelt. Some pragmatic issues in the planning of definite and indefinite noun phrases. In *Proceedings of 23rd ACL*, 1985.

[2] Herbert H. Clark. *Using Language*. Cambridge University Press, Cambridge, England, 1996.

[3] Barbara Di Eugenio and Bonnie Webber. Pragmatic overloading in natural language instructions. *International Journal of Expert Systems, Special Issue on Knowledge Representation and Reasoning for Natural Language Processing*, 9(1):53–84, March 1996.

[4] H. Paul Grice. Logic and conversation. In Peter Cole and Jerry Morgan, editors, *Syntax and Semantics 3: Speech Acts*, pages 64–75. Academic Press, New York, 1975.

[5] Helmut Horacek. Generating inference-rich discourse through revisions of RST-trees. In *Proc. of AAAI*, pages 814–820, 1998.

[6] Pamela W. Jordan, Maxim Makatchev, and Kurt VanLehn. Combining competing language understanding approaches in an intelligent tutoring system. In *Proceedings of the Intelligent Tutoring Systems Conference*, 2004.

[7] Stephen C. Levinson. *Pragmatics*. Cambridge University Press, Cambridge, England, 1983.

[8] Maxim Makatchev and Kurt VanLehn. Analyzing completeness and correctness of utterances using an ATMS. In *Proceedings of Int. Conference on Artificial Intelligence in Education, AIED2005*. IOS Press, July 2005.

[9] Ehud Reiter and Robert Dale. Building applied natural-language generation systems. *Journal of Natural-Language Engineering*, 3:57–87, 1997.

[10] Matthew Stone and Bonnie Webber. Textual economy through close coupling of syntax and semantics. In *Proceedings of 1998 International Workshop on Natural Language Generation*, Niagra-on-the-Lake, Canada, 1998.

[11] Kurt VanLehn, Pamela Jordan, Carolyn Rosé, Dumisizwe Bhembe, Michael Böttner, Andy Gaydos, Maxim Makatchev, Umarani Pappuswamy, Michael Ringenberg, Antonio Roque, Stephanie Siler, and Ramesh Srivastava. The architecture of Why2-Atlas: A coach for qualitative physics essay writing. In *Proceedings of Intelligent Tutoring Systems Conference*, volume 2363 of *LNCS*, pages 158–167. Springer, 2002.

A Schema-based Pedagogical Agent to Support Children's Conceptual Understanding

Zukeri IBRAHIM, Vania DIMITROVA, Roger BOYLE
School of Computing, University of Leeds
Leeds LS2 9JT, UK

Abstract. Researchers acknowledge the difficulty faced by children in understanding new concepts. We propose a Schema Activation and Interpersonal Communication (SAIC) approach where a traditional multimedia system is extended with a pedagogical agent that explains new concepts to a child in one-to-one dialogue that promotes schema-based cognitive tasks. This paper describes the design of the SAIC agent based on both principles derived from the schema theory and dialogue strategies derived from a WoZ study. We also present a prototype that implements the agent in a learning system for teaching children basic Astronomy. An experimental study in real settings with primary school children validated the design of the agent. Initial results of the study suggest that the agent was effective in supporting improvement of the children's schematic knowledge.

Key words: conceptual understanding, schema theory, design architecture

1- Introduction

In classroom settings, teachers support students when they have problems to understand new concepts in a lesson. Helping children is different from helping adults. Explaining new concepts to young children requires supporting the reasoning at their cognitive development stage based on concrete objects and ideas [15]. The support is normally through some dialogue where teachers introduce new concepts by tailoring the explanations to the reasoning ability and the prior knowledge of the children. Studies, e.g. [2], show that computer generated explanations may fail to communicate successfully the meaning of words to children, and miscommunication may occur due to deficiency of appropriate teaching strategies or incapability to address the needs of each individual child. As pointed out by Woolf [18], there is a lack of interactive pedagogical agents that adapt to the children's cognitive development stage. The design of such agents can be based on learning theories that explain how children understand new concepts, as well as on studies of how human teachers support the children's conceptual understanding [5].

The aim of our research is to develop a pedagogical agent capable of engaging in a dialogue to help young children understand new concepts. To inform the design of the agent, we have referred to schema theory [4], [16] that explains how meaning-making occurs and stresses the importance of prior knowledge. Although the schema-based reasoning has been extensively studied and the suitability of the schema theory for the design of learning systems has been acknowledged [14], to the best of our knowledge there are no computational architectures of intelligent tutors based on schema theory. We have proposed an approach for Schema Activation and Interpersonal Communication (SAIC) to support cognitive tasks that occur when a child is learning new concepts through one-to-one interaction with a computer agent [8].

Schema theory is one possible model of the human learning. Other researchers have examined different theories, e.g. ACT* [11], Zone of Proximal Development [13] and constructivism [3]. In this line, our work contributes to computational approaches of intelligent tutoring systems that are derived from learning theories.

There are a number of successful examples of interactive tutors capable of engaging in effective pedagogical dialogue, e.g. [6], [7], [12], to mention a few. However, they have been designed to support university students, while the SAIC approach presented in this paper shows how a pedagogical agent can use dialogue to support children's conceptual understanding by following strategies based on schema theory.

This paper describes the design of the SAIC agent based on both principles derived from the schema theory and dialogue strategies derived from an empirical study with human teachers (Section 2). We also outline the architecture of the agent and describe briefly its implementation in a prototype for teaching Astronomy to children aged seven to eleven (Section 3). An empirical study was conducted to validate the design and examine benefits of the approach. Initial results of the study are discussed in Section 4. Finally, in the conclusions, we point out the contribution of this work to AIED.

2. The Design of the SAIC Pedagogical Agent

Following the methodologies for capturing pedagogical expertise in intelligent tutoring systems (ITS) outlined in [5], to design the behaviour of the SAIC agent we have looked at a theory that explains how people learn new concepts, namely the schema theory, which has been combined with empirical investigation of what dialogue strategies human teachers use to support children's conceptual understanding.

2.1 Deriving design principles from the schema theory

The major claims of the schema theory are examined here to derive principles for the design of the SAIC agent. According to the theory, the knowledge structures in a person's mind are represented in schemas, which are used to understand new information by connecting it to previous knowledge. Learning will be successful when new information is appropriately related to prior knowledge or *existing schemas* [14]. Hence, the first claim followed is:

> **Theoretical claim 1:** *The understanding of a new concept is based on existing schemas.*

Based on this claim, the first design principle for SAIC was formulated as:

> **SAIC design principle 1:** *During the interaction with the child, the agent has to activate prior knowledge and use this knowledge to introduce new information.*

Rumelhart and Norman [16] have proposed three learning processes to account for human reasoning based on schema modification: *accretion* - an existing schema from the prior knowledge is directly used to interpret a new concept, *tuning* - an existing schema has to be slightly changed in order to understand a new concept, and *restructuring* - when existing schema has to be significantly modified to create a new schema that will accommodate the new information. In addition to these, Rumelhart and Norman discuss the *creation* of a new schema when appropriate schema in the prior knowledge cannot be found. Creation is based on stating the definition of the schema and, although very straightforward, rarely leads to meaningful learning. These processes represent a continuous retrieval from the long-term memory, application of previously acquired schemas in new situations, and construction of new schemas [10]. We will call these processes *cognitive tasks* to emphasise that people (children in our case) undertake some cognitive activities to understand new concepts. Therefore, the second theoretical claim we have followed is:

Theoretical claim 2: *The interpretation of a new concept is performed using one of the learning modes: accretion, tuning, restructuring, or creation.*

Based on this claim, the second design principle for SAIC was formulated as:

SAIC design principle 2: *The agent should promote the cognitive tasks proposed by the schema theory.*

Children at the concrete operational stage (mainly aged 7 to 11 years old) think based on concrete objects or ideas [15], as opposed to adults who think based on both concrete and abstract objects or ideas. This nature of children's thinking needs to be considered to effectively support their conceptual understanding. Therefore, the third claim followed is:

Theoretical claim 3: *Children at the concrete operational stage reason based on concrete objects and ideas.*

This led to the third design principle:

SAIC design principle 3: *The agent should provide concrete examples and should avoid the use of abstract concepts.*

While the schema theory provides general guidelines of how to design the agent, it does not give sufficient understanding of what dialogue strategies should be used by a pedagogical agent to promote schema-based cognitive tasks. We, therefore, conducted a Wizard of Oz (WoZ) study to capture strategies employed by human teachers to explain new concepts.

2.2 Deriving dialogue strategies from a Wizard of Oz study

The study was aimed at deriving patterns of dialogue strategies to promote schema-based cognitive tasks. Nine Malaysian primary school teachers, all with significant teaching experience, were asked to explain new concepts to a geographically remote child (in the UK) via a chat-like interface added to a conventional hypertext learning system teaching Astronomy. The child was simulated by a wizard who "created" conditions in which, according to the schema theory, certain cognitive tasks would be required (e.g. schema is known/unknown, schema is created but some properties are unknown, schema is created but the values of some properties are unknown). These conditions triggered dialogue episodes (modelled in SAIC as dialogue games) that corresponded to schema-based cognitive tasks.

Table 1: Dialogue episodes and dialogue strategies

Dialogue episode	**Strategies**
Activation	Asking property value of a schema Asking name of the schema with a property value Showing picture of a schema Informing the ISA category and its instance
Accretion	Asking an instance of the new concept Informing the ISA of the new concept Comparing the new concept with its ISA instances Informing an instance of the new concept ISA Asking whether the student can see the similarity between the new concept and its parent
Tuning	Adding a new property to the new concept which has just been created Informing a wrong property value of the new concept Contrasting the new concept with schemas under the same ISA category Informing the ISA category of the new concept
Restructuring	Comparing and contrasting the new concept with an existing schema

Creation	Informing the property of the new concept Showing a picture of the new concept Informing its ISA category
Introductory	Informing that the lesson is new Informing that the lesson is interesting Informing what the student will learn Informing how the student will learn
Diagnose	Asking whether the student is familiar with the new concept Asking whether the student knows properties of the new concept
Summarizing and confirmation	Confirming the knowledge of the student Informing the properties of the new concept and confirming the knowledge of the student

The dialogue strategies were derived based on an analysis of the dialogue transcripts. The results from the study are discussed in detail in [9], due to space constraints we only present a summary of the dialogue strategies here, see Table 1. These strategies were simulated with the SAIC agent, whose architecture is discussed next.

3. The SAIC Prototype

We propose an architecture (see Fig 1) that extends a traditional multimedia system with a SAIC pedagogical agent which interacts with a student using text or a combination of text and a picture to explain new concepts via one-to-one dialogue with a child.

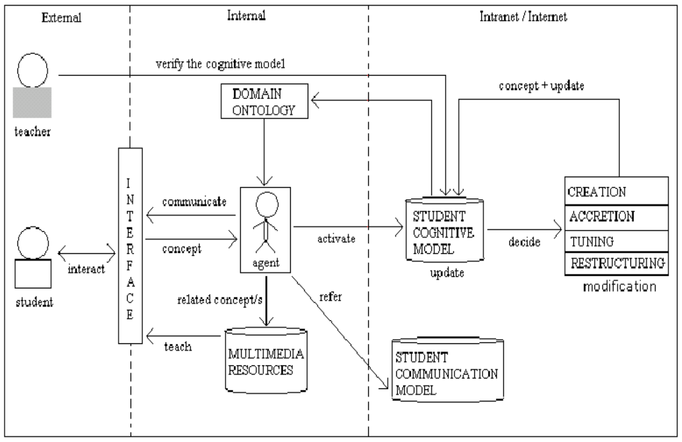


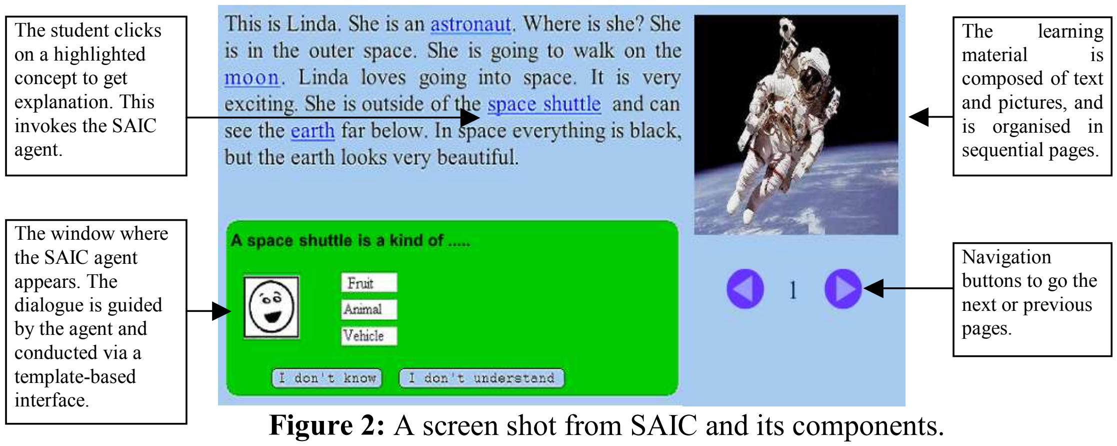
Figure 1: Integration of the SAIC pedagogical agent within a multimedia learning system.

A multimedia learning system presents a sequence of lessons using several media, such as text, pictures, or videos. In addition, the proposed architecture implements the metaphor of an *interactive reading session* where the child goes through some multimedia resources and can ask someone (in this case a computer agent) for an explanation when he/she faces problems to understand new concepts. The SAIC agent is an integrated part of the learning environment.

The SAIC pedagogical agent uses a template-based interface to guide the student reasoning during the interaction (see example in Fig 2). The agent engages the child in a dialogue that employs a student cognitive model that represents the prior knowledge of the student. The cognitive model contains schemas that are learned by the student and is dynamically updated through the interaction with the system. It also contains default assumptions of concrete objects and ideas that correspond to the age group and relate to the abstract domain concepts (e.g. in the study presented in Section 4, the children were assumed to have knowledge of vehicles, animals, seasons, etc. that could be related to basic Astronomy concepts).

The dialogue is organised as a series of dialogue games that trigger corresponding schema-based cognitive tasks. Based on the student cognitive model, the agent selects relevant schemas that have to be activated in order to interpret the new concept. It follows a methodology for assessing schematic knowledge [10] and probes for function (the main action of the concept, e.g. *astronaut function: goes to the moon*), structure ("part-of" relations, e.g. *astronaut-has: protective-cloth*), and process (how the action is performed, e.g. *astronaut-process: pilot-space-shuttle-to-the-moon*). It then selects a suitable schema mode (accretion, tuning, restructuring or creation) and an appropriate explanation strategy (see Table 1). A domain ontology with frame-based descriptions of the main concepts to be learned is used to decide what properties and relationships to include in the dialogue.

A prototype system Going to the Moon was implemented to demonstrate the architecture with the SAIC agent[1]. Basic Astronomy was chosen as a tutoring domain, which contains both concrete and abstract objects and is suitable to illustrate our schema-based approach. A screen shot from the prototype is given in Fig 2. The prototype is implemented in Lingo[2] and follows client-server architecture. The student cognitive model and the domain ontology (built using Protégé[3] and converted to frame-based representation followed in the SAIC agent) are stored on the server. A copy of the system prototype is run on the server to register the child's activities and to record the dialogue.


The student clicks on a highlighted concept to get explanation. This invokes the SAIC agent.
The window where the SAIC agent appears. The dialogue is guided by the agent and conducted via a template-based interface.
The learning material is composed of text and pictures, and is organised in sequential pages.
Navigation buttons to go the next or previous pages.
1
Fruit
Animal
Vehicle
I don't know
I don't understand

This is Linda. She is an astronaut. Where is she? She is in the outer space. She is going to walk on the moon. Linda loves going into space. It is very exciting. She is outside of the space shuttle and can see the earth far below. In space everything is black, but the earth looks very beautiful.

A space shuttle is a kind of .....

Figure 2: A screen shot from SAIC and its components.

To illustrate the interaction with the SAIC agent, we will show an example of how the agent explains the new concept '*space shuttle*'. In the example, the student cognitive model indicates that the schemas '*space shuttle*' and '*airplane*' have been created in previous interactions and from default assumptions, respectively. The domain ontology indicates that, like '*airplane*', '*space shuttle*' is a '*vehicle*' and has a property-value '*has: wings*' which is already in the child's cognitive model. However, the *destination* property of '*space shuttle*' is not in the child's cognitive model. The '*airplane*' schema has a property-value '*destination: airport*' and can be used to tune '*space shuttle*'. Hence, the tuning explanation mode is chosen. A contrasting explanation strategy is initiated that triggers several dialogue games (episodes), as shown in the following dialogue extract. The child is guided by the agent and chooses suggested options (indicated in {}).

Confirmation game
SAIC 1 – "Do you know what the word *space shuttle* means?" //*The agent probes the schema.*
{"Yes", "No"}
Student – "Yes"
SAIC 2 – "You think you know what a *space shuttle* is. Good" //*Confirmation that the schema is created.*
Activation game
SAIC 3 - "A space shuttle is a kind of: …",
{"Fruit", "Animal", "Vehicle". - "I don't know", "I don't understand"}
//*Previous knowledge is activated to probe the relevant group. The interface at this point is shown in Fig 2*
Student – "Vehicle" //*The right category is confirmed.*

[1] The system is available on-line at `http://csw2k017.soc.leeds.ac.uk/saic/`
[2] `Lingo` a scripting language for multimedia authoring.
[3] `http://protege.stanford.edu/`

SAIC 4 - "Brilliant. *Space shuttle* is a kind of *vehicle*. Examples of *vehicles* are: *car, bus* and *bicycle*." // *Confirm-praise and activate relevant schemas. Car, bus, and bicycle are chosen as the student cognitive model indicates that they are the most familiar vehicles to this child.*
SAIC 6 - "*Space shuttle* has *wings*." // *The agent activates the structure of space shuttle from the previous knowledge.*
{" Yes", " No". - "I don't know", "I don't understand"}
Student – "Yes"
Tuning game
SAIC 7 - "The *destination* of an *airplane* is an *airport* but the *destination* of a *space shuttle* is the *moon*." // *Tuning space shuttle by contrasting with airplane.*
{"Continue"}

During the one-to-one interaction, the student can ask for explanations of new concepts, make a selection from the suggested options, and inform the agent when he/she does not know the answer or does not understand the question. Note that the agent avoids directly telling the answer but guides the child's reasoning to reach that answer through schema-based cognitive tasks.

4. Using SAIC in Real Classroom Settings

In order to validate the proposed design architecture and show the effectiveness of the SAIC agent, we conducted an experimental study in real classroom settings. The main objective of the study was to test whether interactions with the SAIC agent in the Going to the Moon system would improve the children's conceptual understanding.

4.1 Experimental Design

An experimental study was performed with 32 students and 5 teachers of Chapeltown Harehills Computer Assisted Learning School (CHALCS[4]) at Leeds. A control group design [1] was chosen where the children were equally distributed according to their reading and writing abilities (as assessed by the teachers) into two groups. The control group used Going to the Moon as a traditional multimedia system without the SAIC agent, while the experimental group used the system with the agent. The study took part in several literacy classes and was organised as a group reading session. Prior to each session the children's schematic knowledge was assessed using word association pre-test. Then, each child was allocated a computer for the reading session. The children with SAIC were briefed that there was an agent that would help them if they needed more explanation. The pre-test scripts were used to assess the students' previous knowledge and initialise the student cognitive models (by hand). A week after the sessions, the children were asked to answer a post-test that was in the same format as the pre-test. Group interviews were conducted to gather the children's opinions about the SAIC agent. The teachers were interviewed about the usefulness and potential of SAIC.

4.2 Results and discussion

Qualitative and quantitative data collected during the study is being analysed. Initial results are presented here. The pre-test and the post-test scripts of each child were compared to examine if there was any improvement in that child's schematic knowledge in terms of function, structure and process (see Section 3). An improvement of the knowledge of a specific concept was indicated if the explanation of that concept in the post-test was more specific, elaborated or complete than in the pre-test. As shown in Figure 3, the students in the experimental group made more schematic improvement than the students in the control group. There was only one child from the group with SAIC who did not make any

[4] CHALCS is a community based activity to provide after school activities for children most of whom come from minority groups. The computers are widely used to assist learning and communication activities are encouraged. In this respect, the SAIC agent was considered favourably by CHALCS staff who fully supported the study and integrated it in their classroom activities.

improvement as opposed to 4 children from the control group who did not improve their schematic knowledge. Some children from the experimental group improved their knowledge of 5, 6 and 7 concepts which was not observed within the control group. This suggests that the SAIC agent can help students explain new concepts presented in a conventional multimedia system. However, a Mann-Whitney test of the improvement numbers of the two groups does not indicate statistically significant difference ($U = 87.000$, $N_1 = 16$, $N_2 = 16$, $p = 0.128$).

To analyse whether the improvement with SAIC was dependent on the students' abilities, we classified the students of the experimental group into three categories according to their reading and writing skills (low, middle, and high). The results are presented in Table 2. A one-way ANOVA test showed that there was statistically significant effect of ability group on schematic improvement in the children who used SAIC ($F_{(2,13)} = 7.276$, $p < 0.05$). This shows that the SAIC agent is more effective for different ability groups at supporting their conceptual understanding.

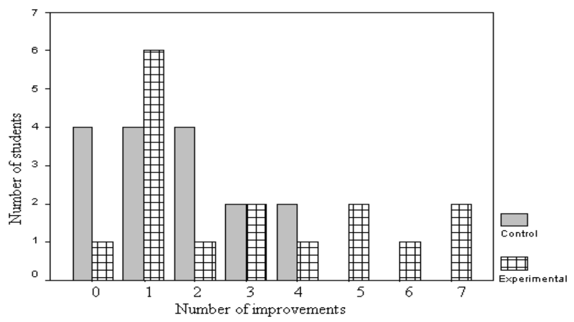
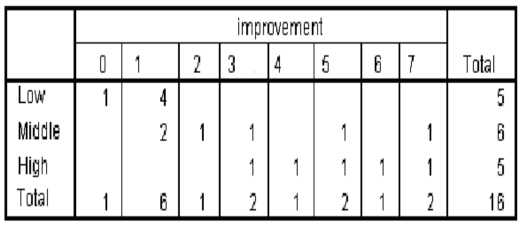


Figure 3: The number of improvements made by the control and experimental groups. Each improvement indicates better schematic knowledge of one domain concept.

Table 2: The number of improvements for each ability category in the experimental group (numbers in the table indicate how many students have demonstrated improvement for 0, 1, 2, 3, 4, 5, 6 and 7 concepts, respectively).



	improvement								
	0	1	2	3	4	5	6	7	Total
Low	1	4							5
Middle		2	1	1		1		1	6
High				1	1	1	1	1	5
Total	1	6	1	2	1	2	1	2	16

We also noted that students with higher number of improvement of schematic knowledge engaged in longer dialogues with the pedagogical agent. The analysis of the dialogues with SAIC showed that most students in the experimental group followed all the explanation steps guided by the agent. During these interactions the agent combined the dialogue strategies, as defined in section 2. This gives some indication that the strategies might have been effective in promoting schema-based cognitive tasks, while further analysis is being undertaken to properly examine this claim.

We have analysed the pre-test and post-test scripts to count correct statements made about isa relation, function, structure and process of each concept. The results show that the students from the control group have indicated more isa links than the students in the experimental group (mean rank 17.53 in control group and 15.47 in the experimental group). At the same time, the group with SAIC could identify more processes of concepts compared to the control group (mean rank 19.28 in experimental group and 13.72 in control group). We noted that the explanations of the experimental group were at a deeper conceptual level but concerned less number of concepts, while the control group wrote about more concepts but included mostly isa relations and functions. These observations correlate with other evaluative studies of pedagogical agents, e.g. [17] that show that dialogues lead to improvement in depth rather than increase in breath.

The students said that they enjoyed working with the SAIC agent and would like to use it in other lessons. They felt that the agent was helping them and was guiding their learning. The teachers supported the use of the pedagogical agent and considered it as helpful to give individualised explanations to each child. However, they also expressed concerns about the effort needed to develop the domain ontology and the initial cognitive model. They were also sceptical about the lack of parental and teacher control. These issues are valid for most intelligent tutoring system and refer to the need of flexibility and learner/teacher control, which can be incorporated in future improvement of SAIC.

5. Conclusions

The SAIC pedagogical agent is an implementation of a novel ITS design architecture based on the principles of schema theory to support children's conceptual understanding by guiding their reasoning in terms of schema activation and modification. We have shown how the claims of schema theory can be followed into the design of pedagogical agent. We also used a Wizard of Oz study to derive teaching strategies of how to effectively perform schema activation and modification in one-to-one interaction with children. A prototype system was developed to demonstrate the approach. An experimental study in real settings with primary school children validated the design of the agent. Initial results of the study suggest that the agent was effective in supporting improvement of the children's schematic knowledge.

References

[1] Ainsworth, S. (2003). Evaluation methods for learning environment. Tutorial for the AIED2003.

[2] Aist, G. (2001). Towards automatic glossarisation: automatically constructing and administering vocabulary assistance factoids and multiple-choice assessment. *Int. J. of AIED*, 12, pp. 212-231.

[3] Akhras, F. N. and Self, J. A. (2000). System intelligence in constructivist learning. *Int. J. of AIED*, 11, 344-376.

[4] Bartlett, F. C. (1958). *Thinking*. New York: Basic Books.

[5] du Boulay, B. and Luckin, R. (2001). Modelling human teaching tactics and strategies for tutoring systems. *Int. J. of AIED*, 12, pp. 235-256.

[6] Freedman, R. (1999) Atlas: A plan manager for mixed-initiative, multimodal dialogue. *AAAI-99 Workshop on Mixed-Initiative Intelligence*, Orlando, FL.

[7] Graesser, A. C., Person, N. K., Harter, D., and TRG (2000). Teaching tactics in AutoTutor. Proceedings of the ITS02 Workshop on Empirical Methods for Tutorial Dialogue Systems

[8] Ibrahim, Z., Dimitrova, V. and Boyle, R. (2003). SAIC: A computational approach for supporting children's conceptual understanding. *AIED2003 Supplemental Proceedings*, pp. 65-70.

[9] Ibrahim, Z., Dimitrova, V. and Boyle, R. (2004). Capturing human teachers' strategies for supporting schema-based cognitive tasks to inform the design of an intelligent pedagogical agent. ITS2004 Workshop on Modelling Human Teaching Tactics and Strategies, pp. 5-14.

[10] Kalyuga, S (2003). Rapid assessment of learners' knowledge in adaptive learning environments. *AIED2003*, pp. 167-174.

[11] Koedinger, K. R. and Anderson, J. R. (1997). Intelligent tutoring goes to school in the big city. *Int. J. of AIED*, 8, pp. 30-43

[12] Lesgold, A., Lajoie, S., Bunzo, M. and Eggan G. (1992). Sherlock: a coached practice environment for an electronics troubleshooting job. *In Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches,* Larkin, Jill, Chabay, Ruth (eds), Lawrence Erlbaum Associates, Hillsdale, New Jersey.

[13] Luckin, R., and du Boulay, B. (1999). Ecolab: the development and evaluation of a Vygotskian design framework. *Int. J. of AIED*, 10(2), pp. 198-220.

[14] Marshall, S. P. (1995). *Schemas in problem-solving*. London: Cambridge University Press.

[15] Piaget, J. (1977). *Epistemology and psychology of functions*. Dordrecht: D. Reidel Publishing Company.

[16] Rumelhart, D. and Norman, D. (1978). Accretion, tuning and restructuring: Three modes of learning. In. J.W. Cotton and R. Klatzky (eds.), *Semantic Factors in Cognition*. Hillsdale, NJ: Erlbaum.

[17] Siler, S., Rosé, C. P., Frost, T., Vanlehn, K., and Peter Koehler (2002) Evaluating Knowledge Construction Dialogs (KCDs) versus mini lessons within Andes2 and alone. ITS02 Workshop on Empirical Methods for Tutorial Dialogue Systems, Spain.

[18] Woolf, B. P. (2003). *Building intelligent tutoring systems*. Book draft.

Text Classification Rule Induction in the Presence of Domain-Specific Expression Forms [1]

Adam Carlson [a], Steven L. Tanimoto [a,2]

[a] *University of Washington, Department of Computer Science and Engineering*

Abstract. We describe a new method for learning text-classification rules from examples. The text consists of messages written by students in an online learning environment, and it may contain ungrammatical expressions as well as specialized expressions such as formulae. The method is based on the version-space machine learning technique. Experiments show that our method successfully generalizes over certain classes of embedded numerical expressions involving ranges of values in RGB triples that represent colors in an image processing system.

Keywords. text classification, learning, numerical expressions, domain-specific terms, rule induction

1. Introduction

When students interact with online learning systems, they generate events sequences that express patterns of activity. Learning environments such as the INFACT system [7] at the University of Washington capture these event sequences and record them in a database where they are available to a suite of tools for analyzing them.

Student activity data collected by systems such as INFACT fall into three categories: (1) textual messages posted by students in dialogues and in response to assignments, (2) sketches drawn to accompany textual messages, and (3) user-interface events that occur as students operate tools such as calculators and programming environments. In this paper, we discuss a pattern recognition problem related to data in category 1. The problem is to take textual content, labeled with categories and induce rules that can classify these and additional, unseen messages correctly into the categories to which they belong.

The text classifications rules we induce are used for two purposes. One is to construct "diagnoses" of misconceptions that teachers can inspect in order to monitor the progress of their students. The other is to automatically construct feedback that can be given to students to help them overcome obstacles and learn more effectively. The INFACT system already includes a facility for manual construction of text classification rules[8]. This works adds the capability to automatically learn the rules.

[1] Research supported in part by the US National Science Foundation under grant EIA-0121345.

[2] Correspondence to: Steven Tanimoto, Box 352350, Dept. of CSE, University of Washington, Seattle, WA 98195, USA. Tel.: +1 206 543 4848; Fax: +1 206 543 2969; E-mail: tanimoto@cs.washington.edu.

Several systems have addressed conceptual classification of general student writing[1, 6,2]. This paper describes a particular variation of the problem of inducing text classification rules. In some cases, natural-language text will include formal or semi-formal representations of structured information, often of a domain-specific nature. For example, program snippets can be viewed as textual representations of parse trees and mathematical equations use a well-defined formalism with a domain-specific interpretation. We call such textual representations *notational text*. We present an architecture for learning textual classification rules that can be specialized to handle notational text. The architecture can be extended to incorporate different kinds of notational text and allow different learning biases. We describe a particular extension of the architecture that learns to recognize expressions that we call *RGB color specifications*. These expressions represent color values in the RGB color space.

We present the learning algorithm in section 2. In section 3, we motivate and describe RGB color specifications as a form of notational text. Section 4 explains the extension to the rule language that allows us to learn RGB color specifications. We describe some initial tests and results in section 5. Finally, we discuss possible sources of error and the potential for broader applicability in section 6.

2. Learning Notational Text Classifiers

Our rule learner uses a variation of the Version Space algorithm [5]. It also uses an architecture inspired by the Version Space Algebra [4,3]. The version space algebra allows for a hierarchically composable rule language. The version space algebra uses multiple components called *primitive version spaces* to learn characterizations of different parts of each training example. For example, there may be a primitive version space that learns to identify keywords in textual input, and another one that looks for textual representations of numbers. These primitive version spaces are then combined by *compound version spaces* that use the primitive version spaces to form more complex stuctures, such as phrases or, in our case, notational text. Compound version spaces can, in turn, be combined, resulting in a rule language tree.

2.1. Non-notational example

We begin by explaining how we use the version space algebra to handle simple sequences of text and numbers and then describe how this mechanism is used to learn RBG color specifications. The version space algorithm maintains the set, S_{cons}, comprised of all hypotheses consistent with all training examples. Because S_{cons} may be exponential or infinite, it is represented implicitly. This is done using a partial order, $\subseteq$, to induce a lattice over possible hypotheses. Frequently the “is more general than” relation is used. Given this lattice, the set S_{cons} can be represented by two *boundary sets*. The specific boundary set B_S contains all the maximally specific hypotheses that are consistent with all training examples and the general boundary set B_G contains all the maximally general hypotheses that are consistent with all training examples. The set S_{cons} is the set of all hypotheses that are at least as general as all members of B_S and no more general than all members of B_G.

If a new labeled training example is already consistent with all hypotheses in the B_S and B_G sets, then it is necessarily consistent with all hypotheses in S_{cons} and no action is taken. However, if the example is inconsistent with any hypotheses in the boundary sets, then those hypotheses must be modified to account for the new example. If a positive example is classified as negative by any hypothesis in B_S then that hypothesis is removed from the set and replaced with the maximally specific generalization of the hypothesis that classifies the example as positive. Similarly, a negative example might require that inconsistent hypotheses in B_G be specialized to correctly classify the example as negative. If any member of the specific boundary set is at least as general as any member of the general boundary set, then the version space is said to collapse. This happens when the data are inconsistent. Our system handles a collapsed version space by removing the example that caused the collapse and using it to start a brand new version space. We call this mechanism *lazy disjunction*, because it creates multiple version spaces, each of which represents a separate disjunct.

Our architecture includes two primitive version spaces. The *term* version space recognizes arbitrary words (i.e. whitespace separated sequences of alphanumeric characters) and the *numeric range* version space recognizes numbers within a range. We use the convention of referring to hypotheses as X(y) where X denotes the version space and y denotes the specific hypothesis, so Term(foo) is a hypothesis of the Term version space that requires the word "foo" to occur in a string. When it is unambiguous, we use initials to denote the version space, so this could also be written, T(foo).

In the version space algebra, the hypotheses of a particular version space can be seen as imposing constraints on examples. A hypothesis of the term version space will classify as positive any string that contains that particular term. Compound version spaces impose the constraints of their constituents and possibly additional constraints. Our system includes the compound version space, *sequence*, which requires that all its constituents appear in a particular order. For example, the hypothesis $h = S(T(foo) \prec T(bar))$ is a hypothesis of the sequence version space that includes two hypotheses of the term version space, $T(foo)$ and $T(bar)$. The hypothesis h requires that these terms occur in the specified order in any example that it classifies as positive.

The version space algorithm begins with the general boundary set equal to $\{\cup\}$, a set containing just the universal hypothesis and the specific boundary set equal to $\{\emptyset\}$, a set containing just the empty hypothesis. Consider a positive training example, "A 10 by 20 rectangle." Because the empty hypothesis fails to correctly classify this example as positive, it must be generalized. The terms that are found in this string are, T(A), T(10), T(by), T(20) and T(rectangle). In addition, the numeric ranges NR(10) and NR(20) are also present. However, the positions matching T(10) and NR(10) overlap, as do T(20) and NR(20). As a result four sequence version spaces are created:

$$S(T(A) \prec T(10) \prec T(by) \prec T(20) \prec T(rectangle)), \tag{1}$$

$$S(T(A) \prec NR(10) \prec T(by) \prec T(20) \prec T(rectangle)), \tag{2}$$

$$S(T(A) \prec T(10) \prec T(by) \prec NR(20) \prec T(rectangle)), \tag{3}$$

$$S(T(A) \prec NR(10) \prec T(by) \prec NR(20) \prec T(rectangle)). \tag{4}$$

Suppose a new positive example, "the square is 15 by 15", is presented to the system. This example only shares one non-numeric term with the previous one, that being "by". However, it does have numbers, and the numeric ranges in sequences (2), (3) and (4) can be generalized to include them. Thus each of the version spaces above is generalized below (in the same order):

1	Red: 255 Green: 255 Blue: 0 to make yellow
2	Gray is made with (200, 200, 200)
3	One can make purple by combining the colors, red and blue
4	Grey can be made by the combination of 200 of red blue and green.
5	150 red, 200 blue, 0 green.

Table 1. Examples of students' definitions of colors

$$T(by), \tag{5}$$

$$S(NR(10-15) \prec T(by)), \tag{6}$$

$$S(T(by) \prec NR(15-20)), \tag{7}$$

$$S(NR(10-15) \prec T(by) \prec NR(15-20)). \tag{8}$$

The system also provides a compound version space called a *conjunction*, which requires the non-overlapping presence of each of its constituents, but imposes no ordering constraint. While the sequence and conjunction compound version spaces are quite general, the idea of composable version spaces allows us to craft learning components that can identify specific forms of notational text. Next, we describe one such form.

3. RGB Color Specifications

One of the settings in which we use INFACT and our text classification system is a class that teaches mathematical concepts using image processing. In this class, students apply mathematical transformations to images and see the results using a calculator-like learning environment called PixelMath. For example, the students may enter a formula like `src(x,y)*2` into the calculator. This formula will double the pixel value of each pixel in a source image to produce a contrast-enhancing effect in the destination image.

The PixelMath class uses online discussion in a web-based forum. Students discuss how they would construct formulae to accomplish various image processing tasks and predict what certain formulae will do to an image. One of the early units in the class is a color matching exercise in which students are asked to describe what red, green and blue pixel values they would use to produce certain colors. This unit is intended to familiarize students with the standard representation of images as a two-dimensional array of numeric triples. It is in the context of this exercise that we first encountered RGB color specifications as a notational form that we wanted our text classifier to recognize.

Some examples of student response are shown in table 1. Students use several formats to describe RGB colors. Example 1 shows a format using a sequence of labeled values. Each color value is preceded by the color name. When this format was used, the order of the colors varied, and sometimes one or more of the colors would be missing. In some cases the letters "R", "G" and "B" are used instead of the full words, "red", "green" and "blue". The second format, seen in example 2, is a simple numeric triple, such as "(100, 250, 200)." This format always included all three numbers and invariably represented the values in the order red, then green then blue. Finally, some students use prose to describe RBG colors, as seen in examples 3 and 4. Finally, some students use a format similar to that of example 1 but in which the color channel names came after the values. We return to this format in section 5.

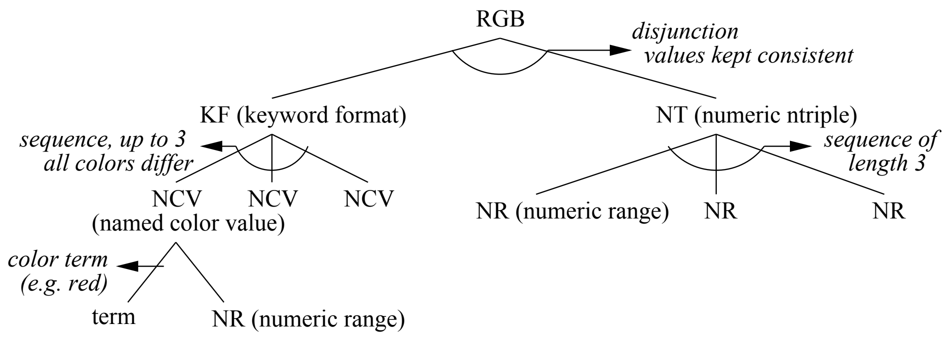


Figure 1. Compound version space tree for RGB color specification. Italicized notation indicates additional constraints imposed by a compound version space on its constituents.

It is important to note that our goal is not to learn the format of RGB color specifications, but rather to learn misconceptions students may have about RGB colors. For example, we wanted to know if students thought that the RGB values were percentages, which would result in them always using values between 0 and 100.

4. RGB Color Specification Version Space

The idea of compound version spaces farming out some of their work to other version spaces and then imposing additional constraints is used in the handling of RGB color specifications. Color specifications may occur in two formats, each having its own compound version space representation. The *numeric triple* (NT) version space is simply a sequence version space that requires that there be exactly three constituents, all of which are numeric ranges. The *keyword format* is a sequence of three instances of an intermediate compound version space, the *named color value* (NCV). The NCV version space, in turn, recognizes a single "colorname numeric-value", and is implemented as a two-element sequence in which the first element is a term matching one of, "red", "blue", "green", "r", "g" or "b". The second element must be a numeric range. The *keyword format* (KF) version space imposes the additional constraint that no color name is repeated. Finally, the *RGB color specification* compound version space is a disjunction between one *keyword format* version space and one *numeric triple* version space. The *RGB color specification* enforces the further constraint that the numeric ranges for each color value are identical between the two formats. Thus the hypothesis $RGB(0-10, 100, 200-255)$ is a disjunction between the hypotheses $\mathrm{KF}(\mathrm{NCV}(red, 0-10), \mathrm{NCV}(green, 100), \mathrm{NCV}(blue, 200-255))$ and $\mathrm{NT}(0-10, 100, 200-255)$. The version space algebra tree for RGB color specifications is shown in figure 1.

5. Tests and results

To test the system, we collected data from the online discussion group software, INFACT-FORUM, consisting of student responses to an additive color-mixing assignment. Students were asked to describe what color values they would use to obtain particular target colors, such as yellow, pink and grey. Each example was manually classified as positive

	True+	True−	False−	False+	Precision	Recall	F-Score
Yellow	22	0	0	43	0.338	1.000	0.506
Purple	12	12	3	38	0.240	0.800	0.369
Pink	19	12	3	31	0.380	0.864	0.528
Grey	6	0	0	59	0.092	1.000	0.169
Total	59	6	6	171	0.257	0.908	0.400

Table 2. Results for RGB only unpruned data sets.

	True+	True−	False−	False+	Precision	Recall	F-Score
Yellow	6	2	2	18	0.250	0.750	0.375
Purple	0	3	2	23	0.000	0.000	0.000
Pink	16	12	0	0	1.000	1.000	1.000
Grey	0	26	2	0	0.000	0.000	0.000
Total	22	43	6	41	0.349	0.786	0.484

Table 3. Results for Term only pruned data sets.

or negative for the color requested depending on whether the color values the student chose produced a reasonable shade of the requested color. It turned out that almost all examples were positive, so we use positive examples of each color as negative examples of every other color.

Because the data were sparse, we used a leave-one-out testing regime. For each color, we considered each example e_i in turn. We trained on all other examples to learn a classification rule and then tested the classification of example e_i. This can be thought of as k-fold cross-validation with $k = n$. Two kinds of data were used: pruned and unpruned. The pruned data consisted only of examples including RGB terms in the two formats handled by the RGB recognizer. The unpruned examples also included other patterns. The data were pruned by hand.

With the unpruned data, the precision was fairly low, due to the large number of false positives obtained due to overgeneralization of the rules on the basis of misinterpreted RGB values. With the pruned data, the use of the domain-specific RGB component tended to improve precision over the use of straight text terms only; RGB alone gave a 0.50 total precision value compared with 0.35 for term only. Combining the two gave an even better result: 0.66. Recall remained relatively unchanged.

The system performed better on the colors yellow and pink than purple and grey. Using pruned data the system was never able to generalize properly for purple and grey. Part of the reason for this is that the RGB color specification version space doesn't do any reasoning about the relationship between numerical ranges for different color channels. It is unable to learn a rule like "all three color values must be the same." So if the system sees a positive example where a student created a very dark grey, such as "grey is (50, 50, 50)" and a positive example with a light grey, such as "I made grey with (200, 200, 200)", then it will learn the overly general hypothesis RGB(50-200, 50-200, 50-200).

Another source of difficulty in classifying purple was the use, by a few students, of a format very similar to the keyword format described in section 4. These students used three named color values, but rather than writing "name: value", they put the value first and then the name. (See example 5 in table 1.) This example confused the rule learner since it interpreted it as a color specification in which the red value is 200 and the blue value is 0.

	True+	True−	False−	False+	Precision	Recall	F-Score
Yellow	6	9	2	11	0.353	0.750	0.480
Purple	0	26	2	0	0.000	0.000	0.000
Pink	16	1	0	11	0.593	1.000	0.744
Grey	0	26	2	0	0.000	0.000	0.000
Total	22	62	6	22	0.500	0.786	0.611

Table 4. Results for RGB only pruned data sets.

	True+	True−	False−	False+	Precision	Recall	F-Score
Yellow	5	9	3	11	0.313	0.625	0.417
Purple	0	26	2	0	0.000	0.000	0.000
Pink	16	12	0	0	1.000	1.000	1.000
Grey	0	26	2	0	0.000	0.000	0.000
Total	21	73	7	11	0.656	0.750	0.700

Table 5. Results for combined pruned data sets.

6. Discussion

The rule learner is able to successfully classify examples for some colors, though it has trouble with others. In particular, it fails when the restriction bias inherent in the RGB color specification version space doesn't match the actual concepts that should be learned. This mismatch can happen in two ways. First, if the format of the color specifications varies significantly from that described by the version space, as when students reversed the order of color names and values. Second, if the concept that describes a positive example involves a relationship between the values of different color channels that the version space can't represent. The concept that best describes RGB values for the color grey is that all three color channels are equal. But our RGB color specification learner can't represent numeric relationships between channels, only ranges of values for each channel independently.

These observations give us some insight into how the method described in this paper could be used for other types of notational text, what limitations might be encountered, and some ways to extend it for greater applicability. We will sketch a similar approach that can form hypotheses describing mathematical expressions, and then discuss how such a system may work.

To construct a version space algebra for mathematical expressions, we would start with primitive version spaces that identify simple mathematical terms, numbers and variables. We could use the numerical range version space described above for numbers, and the *variable* version space would be similar to the term version space. We would then construct a compound version space for binary operators. It would be a sequence of three elements, the first and last would be expressions and the middle one would be one of the accepted operators. Similar compound version spaces would handle parenthesized expressions and equations. Finally, the expression version space would be a disjunction between the various forms of a valid expression. The application of this method to mathematical expressions is just one possible domain. Any formal notation that has a tree-like structure would be representable in our system.

While the system sketched above could learn hypotheses describing general mathematical expressions, it's clear from our experience with RGB color specifications that additional biases would need to be included to learn effectively. First, it would probably be necessary to restrict the variable version space. It might be useful to limit length of variable names or even have a hard-coded list of commonly used variable names for a particular domain. Second, it would probably improve the ability to learn useful classification rules if the version space components are able to analyze the relationships between their constituents. To make this feasible, it would make sense to have specific variants of the version spaces that capture particular common misconceptions associated with mathematical expressions, such as dropping a negative sign before a term or doing multiplication instead of division. In general, we believe that the system will perform better whenever more domain knowledge can be incorporated into the version space representations of the notational text.

In addition to pointing up the need for more problem-specific and domain-specific biases in the version spaces, our experiences have led us to consider possible extensions to the algorithm. Because the lazy disjunction mechanism is order-dependent, it is prone to learning disjunctions that represent coincidental similarities and differences between examples. We believe that disjunctive concepts could be better learned using a two-pass approach in which examples are first clustered into sets, each of which represents a separate disjunct. Each disjunctive set could then be used to train the system separately. However, we are not yet sure how to cluster examples of notational text.

Another problem we encountered was the system's ability to recognize notational text amid non-notational prose. This was the reason the system performed so poorly on unpruned data. A pre-processing step that can recognize and delimit examples of notational text, based on the version space definitions would be of great benefit. Then the notational text learner could be applied to appropriate sections of text, and other mechanisms could be applied to the rest.

References

[1] Adam Carlson and Steven L. Tanimoto. Learning to identify student preconceptions from text. In Jill Burstein and Claudia Leacock, editors, *HLT-NAACL 2003 Workshop: Building Educational Applications Using Natural Language Processing*, pages 9–16, Edmonton, Alberta, Canada, May 31 2003. Association for Computational Linguistics.

[2] Arthur C. Graesser, Peter Wiemer-Hastings, Katja Wiemer-Hastings, Derek Harter, Natalie Person, and the Tutoring Research Group. Using latent semantic analysis to evaluate the contributions of students in autotutor. *Interacting Learning Environments*, 2000.

[3] Tessa Lau. *Programming by Demonstration: A Machine Learning Approach*. PhD thesis, University of Washington, 2001.

[4] Tessa Lau, Steven Wolfman, Pedro Domingos, and Daniel Weld. Programming by demonstration using version space algebra. *Machine Learning*, 2003.

[5] Tom Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[6] C. P. Rosé, A. Roque, D. Bhembe, and K. VanLehn. A hybrid text classification approach for analysis of student essays. In *Proceedings of the HLT-NAACL Workshop on Educational Applications of NLP*, 2003.

[7] Steven Tanimoto, Adam Carlson, Justin Husted, Earl Hunt, Josef Larsson, David Madigan, and Jim Minstrell. Text forum features for small group discussions with facet-based pedagogy. In *Proceedings of CSCL'02, Boulder, CO.*, 2002.

[8] Steven Tanimoto, Susan Hubbard, and William Winn. Automatic textual feedback for guided inquiry learning. In *AIED 2005*. IOS Press, 2005.

The Uses of Code Fragments in Programming Tutorials

Mary McGee Wood
School of Computer Science, University of Manchester

Abstract. In human-human, keyboard-to-keyboard programming language tutorials, fragments of code are mixed with natural language in explanations of programming concepts and constructs. I describe a series of 37 distributed learning tutorial dialogues on the C programming language, held world-wide over four months, and propose an annotation scheme for the uses served by code fragments. A preliminary analysis of the data set is presented and directions for further research are explored.

Keywords. programming languages, human-human dialogue

1. A corpus of programming language tutorials

Mixed language explanations in learning environments include code fragments embedded in natural language in programming tutorials. Distributed learning courses in C and Java programming, taught world-wide by the Teaching Innovation Unit in the School of Computer Science at the University of Manchester, are supported by on-line, human-human, keyboard-to-keyboard tutorials held in a chat-room environment. Students may be taking single modules to update their professional skills, working towards a modular degree of MSc, or competing for admission to a full-time residential MSc programme. They are sent course material and assessed exercises: the tutorials are optional, and designed for the students to turn up and ask for help when they need it. Thus the "task initiative" (Chu-Carroll & Brown 1998, Core et al 2003) lies largely with the students rather than the tutors, unlike most of the tutorial dialogues which have been studied. These tutorials are logged, and the logs made available for reference to the tutors and students on each course.

For the present study, the logs of a series of 37 tutorials on the C programming language, held from October 2002 to February 2003, were chosen as a data set. Three tutors and 18 students are represented, although a typical tutorial involves one tutor and three or four students. Tutorials typically last for one hour. The dialogues were annotated for the uses served by the code fragments, and the results analysed.

It is worth stressing that this is "live" data. There are no questions about experimental design, because there are no experiments. The data we have is messy and complicated, but gives an accurate snapshot of how (some) tutors and (some) students really use language in general, and code fragments in particular, in a serious learning environment.

The 37 logs in the chosen set were lightly cleaned up (purely by removing system messages: all user input was untouched apart from anonymisation). The sections which included code fragments were isolated by hand (for now: automatic recognition of code fragments should be tractable). Annotation with the ten categories of use explained below was also done by hand, by one annotator (the author). (For now: an obvious research priority is to cover more data with more annotators.)

The nature of the chat-room environment means that user input appears as separate lines: a new line appears on the screen at some indeterminate time after a user closes and sends it by typing Return[1], and begins with the time received and the system name of the sender. In the context of the dialogues overall, I refer to these as "turns"; when specifically discussing code, I refer to "lines". While natural language turns are usually segmented along natural syntactic / semantic / pragmatic boundaries, lines of code are usually segmented by the syntax and/or formatting conventions of the programming language. Thus it is common for a conceptual unit to appear spread over several lines: I refer to these units as "fragments". For example,

```
<T> ok, but what if I do the following:
<T> char first[20];
<T> char second[20];
<T> if (first == second)
<T> is this still using pointers?
```

includes three lines of code, which form one fragment.

2. An annotation scheme

The first obvious distinction to make, in analysing code fragments, is between their use by tutors and by students. Bottom-up analysis of the data shows three functions which code fragments proper can serve: asking a question, answering a question, or quoting a line or fragment to comment on. Program output, and command line input, are also significant formal language elements.

2.1. Tutors

A good tutor uses code fragments in all these ways.

TQn - Tutor question. A tutor uses a line or fragment of code in asking a question, often to test a student's understanding of a programming construct or

[1]The Student Answer example below shows one typical effect of this. Bear in mind that, while some students may be using local ethernet connections in Manchester, others may be working down noisy telephone lines from Latvia or South Africa.

a higher concept.

```
<T> ok, so if I do:
<T> int i = 4;
<T> int *int_ptr = i;
<T> is that ok?
```

TA - Tutor answer. A tutor uses a line or fragment of code in answering a question or giving a hint.

```
<T> try -
<T> int i = 4;
<T> int *int_ptr = &i;
<T> read the & as the 'address of' operator, and this reads:
<T> assign to int_ptr the address of i. Ok?
```

There is scope for further sub-division here between giving a hint and supplying a direct answer, but that lies beyond my immediate scope.

TQt - Tutor quote. A tutor quotes a line or fragment of code supplied by a student, in order to comment on it or correct it. Very often followed immediately by an answer fragment giving a correction.

```
<T> (&(info.Letters_in_database) == &(info.Length_of_database)) should
be
<T> (info.Letters_in_database == info.Length_of_database)
```

TO - Tutor output. A tutor quotes output, compile-time (as here) or run-time (the difference is programming-language-specific), either to give a hint as to what is wanted from a student, or as part of debugging.

```
<T> ouch...
<S> ??
<T> tree.c: In function 'add_less_than':
<T> tree.c:79: warning: assignment from incompatible pointer type
<T> tree.c:81: warning: assignment from incompatible pointer type
<T> ... etc....
```

TI - Tutor input. A tutor specifies what needs to be input to the environment in order to compile or run a program. (This maps to the DAMSL (Core & Allen 1997) Information Level category of "task management", whereas our other four uses are all "task".)

```
<T> you do gcc -Wall -pedantic -ansi file.c -o file
```

2.2. Students

The students in our data set show most of the same uses.

SQn - Student question. A student uses a line or fragment of code in asking a question, usually asking for low-level debugging.

```
<S> okay, so inputting a statement like that : strcpy(dest, source);
<S> would that cause an error??
```

SA - Student answer. A student uses a line or fragment of code in answering a question.

```
<T> strcmp means 'string compare'
<T> any ideas on how to use that to test if first is the same as second?
<S1> strcmp(first,second)
<T> use an 'if' here.
<S2> strcmp (first==second)
<T> Gs, on the right track, just needs an if to test the return value
of strcmp
<S1> if strcmp(first,second)=0; printf("first = second")[2]
```

SQt - Student quote. A student quotes a line or fragment of code supplied by a different tutor or the course material.

```
<S> T2 suggested that the following would be better typedef enum white,
back colour;[3]
```

SO - Student output. Not found in this data set, but included as logically possible.

SI - Student input. Here the student is typically asking for help in exactly how to run a program in a particular environment.

```
<S> my emacs didn't recognise the command -wall. so i compile my progs.
without it. jus -gcc -ansi -pedantic...
```

3. Some results

The 37 dialogues contain 7284 turns, an average of 197, with a range from 28 to 359. Code-lines total 884, or 12%, with a range from 0 to 27%. The dialogues with no code are either short exchanges with little or no technical content, or, more interestingly, show a deliberate tutoring strategy of hinting rather than giving

[2]A good example of "lag": S2's suggestion was typed before seeing the tutor's "if" hint which appears before it in the log.

[3]The student really did type "back" for "black".

answers:

```
<S> Ok another question:
<S> according to the requirements we have to ensure that each part
the 'Posted date' information is held as a separate field.
<S> What are these parts? (month, day, year, time) Should we use nested
structure?
<T> you definately need month, day, year, time, but should also consider
an AM/PM field
<S> Ok thanks
<T> this struct should be typedefed to be a new type that is
<T> useable in the rest of the struct, so yes, a nested structure that
is clarified by typedefed struct
<T> also consider using typedefed structs for other fields.
```

In fact it is a peculiar feature of these dialogues that they attempt to *avoid* using code, because the tutorial logs could give away answers to students further behind in the exercises than those present at the time: there are numerous examples of tutors explicitly refusing to contribute real code, or urging students not to. It will be interesting to study the parallel uses of pseudo-code and natural language paraphrase.

Tutors' code-lines nevertheless outnumber students' by 755 to 129, a factor of 5.8. However tutors' fragments are significantly longer than students', especially for answers, which average almost three times as many lines (the table shows average lines per fragment for each category). Therefore tutors' fragments outnumber students' by a factor of only 3.5 (349 / 94).

	Qn	A	Qt	O	I
Students	1.62	1.1	2	-	1
Tutors	2.08	3.12	2.06	4.93	1.58

The distribution of categories across the time span of the series is also interesting. Discussions of system input are relatively common in the first few tutorials, while students are learning how to use the environment, and then pretty much vanish. Discussions of system output are rare early on, and typically take the form of desired run-time output used as a hint as to what is wanted in a particular exercise:

```
<T> Hmm.. does this help answer the question?
<T> int i;
<T> for (i = 0 to 9) {
<T> printf("%c ", i + '0');
<T> }
<T> will print
<T> 0 1 2 3 4 5 6 7 8 9
```

They then disappear completely only to reappear, densely, towards the end of the course when the tutor is helping the students to debug their code, as in the typical compiler output example in section 2. It is also interesting, and perhaps unexpected, that the use of code in questions largely disappears as the series progresses.

4. Future work

Narrowly directed, the first priority for future work is clearly to look at a larger data set. Many aspects of the questions discussed here are highly specific to either programming languages or tutors. A complete series of Java tutorials, with a different tutor, is our next target for analysis. Although the proposed annotation scheme is fairly clear and objective, it is also a priority to bring in additional annotators and look at their degree of agreement.

We will then be in a position to ask how far the proposed annotation scheme carries over to other phenomena and data sets. The "output" and "input" categories clearly are specific to the programming task. However, I would expect the question / answer / quote classification, as far as it goes, to generalise gracefully to similar data such as the proof fragments described in Horacek & Wolska (2004). If experiments with this prove successful, we should go on to consider what finer-grained analysis is possible and useful.

However the data suggests a number of avenues for wider exploration. These include:

- A systematic analysis of the natural language immediately surrounding code fragments. We find passages of fine-grained mosaic:

```
<T> ok, so ctype.h provides a func called:
<T> int islower(int ch);
<T> basically returns true if the parameter ch is a lowercase letter
<T> you could instead of using this func do the following:
<T> if (ch>='A' && ch<='Z')
<T> sorry, meant:
<T> if (ch>='a' && ch<='z')
<T> for lowercase.
```

Tutors (as an impression, not yet quantified) usually introduce fragments explicitly, while students often do not:

```
<T> in C we'd use the loop that most of the examples rely on,I.e.:
<T> char ch;
<T> while(ch = fgetc(stdin), !feof(stdin) )
<T> ...
<T>
<T> do you understand this loop?
<S> is it not int ch; ?
```

Anaphoric reference into a fragment is particularly difficult to resolve: in the first of these examples "this" refers to the complete fragment, while in the second "that" probably refers to a single function name:

```
<TG> ok, but what if I do the following:
<TG> char first[20];
<TG> char second[20];
<TG> if (first == second)
<TG> is this still using pointers?

<TG> ok, so how about the following:
<TG> I want to see if the following 2 strings are the same:
<TG> char* first = "first";
<TG> char* second = "second";
<TG> there's another func with the following sig:
<TG> int strcmp(const char* ch1, const char* ch2);
<TG> strcmp means 'string compare'
<TG> any ideas on how to use that to test if first is the same as second?
```

However even a human domain expert consulted on this was not completely confident about the second judgement, so it is unlikely that automatic anaphora resolution will be terribly successful in the near future. But it is important in detecting and addressing student misconceptions.

- A systematic analysis of dialogue strategies. How, in particular, are multi-party conversations conducted keyboard-to-keyboard? There are many details to be considered here, especially as regards turn-taking and initiative. It will be interesting to compare our data and analysis with other work such as Jordan & Siler (2002) and Shah et al (2002).

The inclusion of time-stamps on each line in the logs allows us to look at the timing of turns: while the multi-party exchanges can be frantic, one-to-one tutorials often show pauses of as much as three or four minutes - usually while the tutor waits for the student to answer a question - which designers of tutorial systems might wish to consider.

- A systematic analysis of tutoring strategies. Even from this sample, we see that a third tutor, who is much less experienced than the two quoted here, seldom uses code fragments, and then only large fragments used as answers; T1 and T2 are much more subtle and flexible. Distinguishing "answers" from "hints" will also be important here. We cannot quantify learning gain as can be done in experiments, but we can certainly study the patterns of student response to different styles of tutoring at a finer grain. Apart from its inherent paedogogic interest, real data on human-human keyboard-to-keyboard tutoring strategies should be essential to the development of keyboard-based human-computer tutorial systems.

- On-line communities are a current focus of attention in the Information Retrieval community (Leuski 2004). These dialogues show a group of diverse people working together on a significant learning task over several months, and so give us real data on the evolution of groups and individuals, both in general, and specifically in a learning environment.

- Corpus collection and management in a real-world working environment presents different issues from those involved when generating data through experiments. I started with a tar file holding two years' worth of system logs, comprising 771 logs from 14 courses (including Bioinformatics and Object-Oriented Design, as well as the programming courses), which yielded 318 worthwhile dialogues. Even those contained large quantities of system messages and other rubbish, raising questions of what to take out and leave in. After that, anonymisation was a non-trivial question, as the participants often share personal information which cannot (easily, if at all) be automatically detected. Only after this is done can we begin to select appropriate subsets of the corpus for the study of the many different questions which can be explored.

References

Chu-Carroll, J. & M. K. Brown. 1998. *An Evidential Model for Tracking Initiative in Collaborative Dialogue Interactions.* **User Modeling and User-Adapted Interaction** 8:215-254.

Core, M., J.D. Moore,& C. Zinn. 2003. *The Role of Initiative in Tutorial Dialogue.* Proceedings of EACL.

Core, M. & J Allen. 1997. *Coding Dialogs with the DAMSL annotation scheme.* AAAI fall symposium on Communicative Action in Humans and Machines.

Horacek, H. & M. Wolska. 2004. *Interpreting semi-formal utterances in dialogs about mathematical proofs.* In F. Meziane and E. Métais, editors, **Natural Language Processing and Information Systems**, volume 3136 of LNCS, pages 26-38.

Jordan, P. & S. Siler. 2002 *Student Initiative and Questioning Strategies in Computer-Mediated Human Tutoring Dialogues.* ITS Workshop on Empirical Methods for Tutorial Dialogue Systems.

Leuski, A. 2004. ACM SIGIR 2004 Workshop on *New Directions For IR Evaluation: Online Conversations.*

Shah, F., M. Evens, J. Michael, & A. Rovick. 2002. *Classifying Student Initiatives and Tutor Responses in Human Keyboard-to-Keyboard Tutoring Sessions.* **Discourse Processes** 33(1):23-52.

Graphic and NLP Based Assessment of Knowledge about Semantic Networks

Rainer LÜTTICKE
Intelligent Information and Communication Systems, Department of Computer Science
FernUniversität in Hagen, Universitätsstr. 1, 58084Hagen, Germany
rainer.luetticke@fernuni-hagen.de

Abstract. This paper presents an assessment of learners' knowledge about semantic networks using the Multinet paradigm. Since users of Multinet have to learn the transformation of natural language sentences in semantic networks as well as the reformulation of semantic networks in natural language two kinds of assessment are developed. In the first assessment learners build semantic networks in a graphical tool. The graphics are automatically analysed by comparison with a reference solution. In the second assessment the natural language reformulation is automatically transformed in Multinet and then compared to a reference answer. Depending on the analyses a detailed feedback is presented to the learner for the improvement of his solution.

Introduction

We have developed an virtual laboratory (VILAB[1]) for computer science at the open and distance university in Hagen focused on problem solving [4]. The access to the VILAB server operated at our university is managed by decentral user clients via Internet. Within the lab hypertext (e.g. problems) as well as documents from the WWW can be selected and displayed in a browser. Additionally, complex graphical software-tools installed on the server can be activated. The problems given in VILAB are divided into different domains of computer science (e.g. relational data bases, programming, natural language processing [NLP]). The learner's solutions for these problems can automatically be analysed and a tutoring component immediately gives feedback for improvements [3]. After this the learner modifies his solution and a further analysis starts, etc.

This paper presents the mechanisms for analysing solutions in the field of semantic networks using the Multinet[2] paradigm [2]. Multinet has been developed as a semantical representation for information given in natural language. The core of the Multinet representation is a semantic network which is formally a directed labeled graph. Nodes in the graph represent certain entities in the discourse area while edges express semantic relations between the nodes. Every possible concept in the real world is eligible as a net node. Inner nodes represent complex concepts. There exists a fixed set of about 110 relations labeling the edges. Each relation has its own pre-defined meaning (e.g. (AGT e a): *a* is agent in an event *e*). Further more sophisticated Multinet concepts are several attributes of the nodes and edges. Besides the structural information given by the net the conceptual representives of Multinet are characterised by embedding the nodes into a multidimensional

[1] Homepage: http://pi7.fernuni-hagen.de/vilab/, Guided Tour: http://inflabor.fernuni-hagen.de/tour_en/
[2] Multilayered extended semantic networks

space of layer attributes (for details s. [2]). For the assessment of knowledge about Multinet we have realised two approches (Sect. 2). The analysis of a semantic network built by a learner in a graphical software tool and the analysis of natural language reformulation of a semantic network in form of Multinet. Thereby this paradigm itself is used for the analysis of this reformulation.

1. Assessment of Knowledge about Semantic Networks

1.1 Analysis of Semantic Networks

Using the graphical software tool MWR[3] the learner has to design in different exercises semantic networks in Multinet form to a given natural language sentence (s. Figure 1). At any time he can start the automatical correction module which is integrated in MWR. This module, written in the Lisp dialect Scheme, compares by logic inferences the linearised semantic network of the learner's solution with the author's reference semantic network. Outcome of this module are entities and relations, which are wrong and which are missing, and what relations do not connect the right entities. The feedback is graphically and textually given. Wrong or unverified parts of the learner's network are marked in red and verified parts in green in MWR. Additionally, the result of the analysis (or depending on the user model only selected parts of it) are textually presented in a hypertext document (e.g. Figure 2 and 3). This feedback can also be enriched by further support hints (e.g. literature or examples, for details s. [3]).

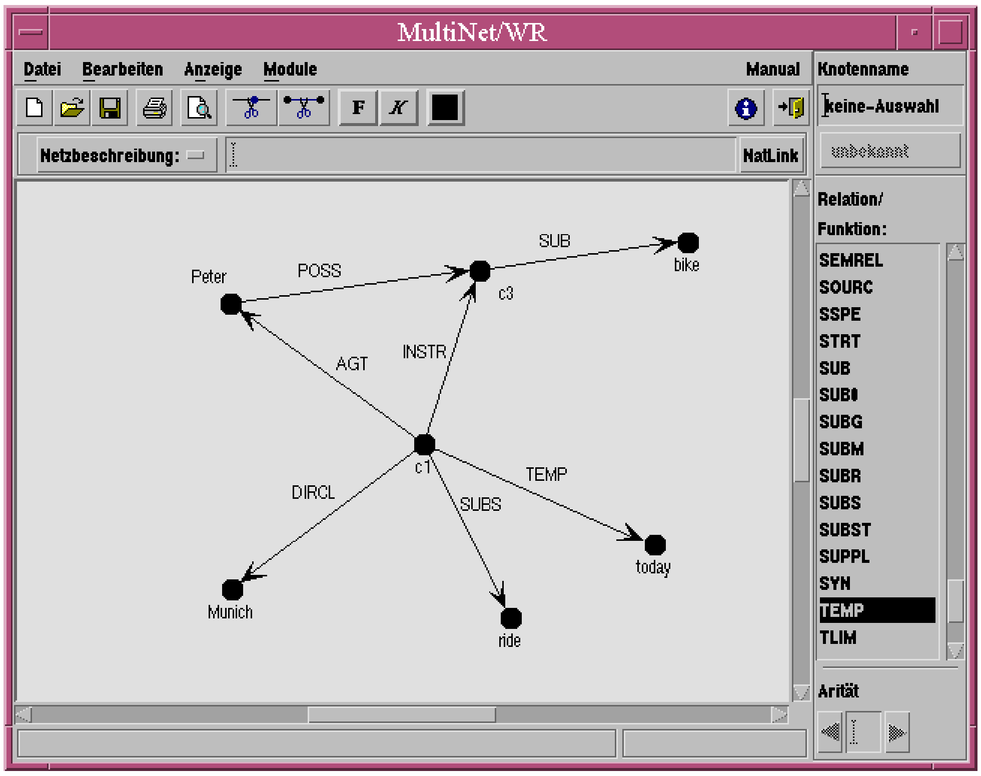


Figure 1: Design of the semantic network to the given sentence "Today Peter rides his bike to Munich.".

[3] **M**ultinet **W**o**r**king Bench is a graphical tool for representation and edition of semantic networks in MultiNet form.

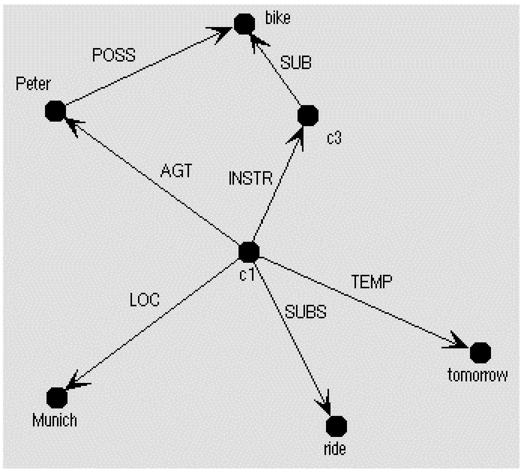
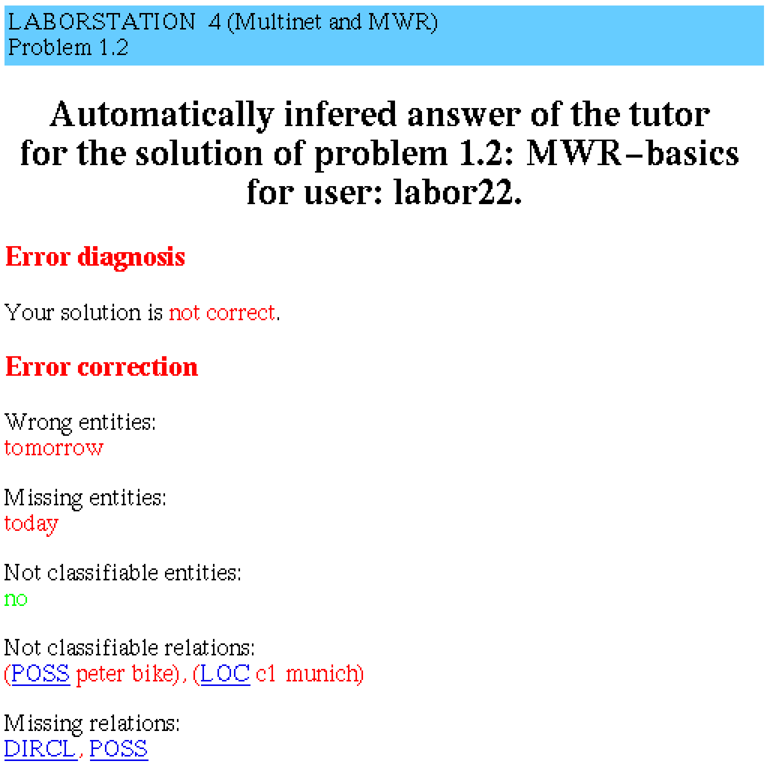


Figure 2: Wrong design of the semantic network to the given sentence “Today Peter rides his bike to Munich.”.


LABORSTATION 4 (Multinet and MWR)
Problem 1.2

Automatically infered answer of the tutor for the solution of problem 1.2: MWR-basics for user: labor22.

Error diagnosis

Your solution is not correct.

Error correction

Wrong entities:
tomorrow

Missing entities:
today

Not classifiable entities:
no

Not classifiable relations:
(POSS peter bike), (LOC c1 munich)

Missing relations:
DIRCL, POSS

Figure 3: Example for (a part of) an (hypertext-)answer of this wrong design with full information of the tutoring component and with literature links to wrong relations. Depending on the user model it also possible that only missing and wrong entities and no information about edges are given, or no links are given, or only the number of wrong edges.

1.2 Analysis of Natural Language Solutions

To a given semantic network in form of Multinet the learner has to write the reformulation in a special text field of MWR (s. Figure 1: field between “Netzbeschreibung” and “NatLink”). After he has automatically transformed his answer in Multinet using WOCADI (formerly: NatLink) [2] by clicking the Button “NatLink” in MWR he can start the request for correction of his textual solution. Then the previously described module compares the learner's transformed answer with the author's reference reformulation (given also in Multinet). The results of this analysis (wrong, missing, unverified or verified Multinet terms) are transformed back in natural language to present mistakes and missing information of the reformulation. While information about the entities can be directly included in the feedback information about relations and connections of relations with entities has to be transformed by rules and templates in natural language phrases or sentences.

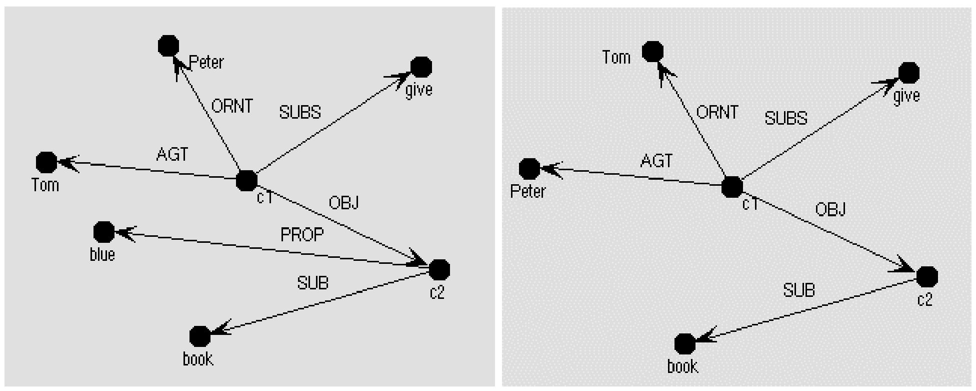


Figure 4: For this given semantic network on the left the reference reformulation would be: "Tom gives Peter a blue book.". The wrong reformulation of this network by a learner "Peter gives Tom a book." would lead to the network on the right.

Figure 4 gives an example for a reformulation exercise. The wrong reformulation in this example would lead to the feedback of the tutoring component including the statements:
"The entity 'blue' is missing in your answer."
"The orientation of the action is not correctly expressed".
"The agent in your answer is wrong." (because (AGT c1 Tom) is part of the reference answer and (AGT c1 Peter) is part of the learner's answer).
For all relations we have such rules and templates which are used and filled with values for the feedback generation.

2. Use the Assessment

Since 2002 we successfully use the correction module of semantic networks (Section 1.1) in practical courses, seminars, and for exercises of courses. Comparing the performance of students having learnt with and without the automatical assessment modules we reveal that the students worked with our interactive learning environment have more exercises correctly solved, have a better understanding of Multinet, and get better marks in exams in this field. Since we have supported with our module for the analysis of semantic networks only the direction from "natural language text" to "semantic network" we have implemented in 2005 the module for the other direction (Section 1.2). The first use of this module will take place in the summer term 2005 in a practical course about NLP and we hope that we can further improve the performance of the students.

Our analysis of natural language reformulation is a kind of short answer assessment for a special domain. This assessment is limited to exercises in which an ordinary semantic network is given and the learner has to express the information of the network in one natural language sentence in German[4]. The automatical transformation in general as well as the correction modules are successful in the most cases. However, the transformation in Multinet is only possible for one sentence because several sentences had to be automatically assimilated to one network. We are working on this task but our currently success rate is not high enough using automatical assimilation in regular teaching.

3. Perspective

We have the perspective that in the future we can enlarge our short answer assessment from the reformulation of semantic networks to open questions in different domains and to answers consisting of several sentences (essays). For this aim we work on the automatical assimilation and on the comparison between the nets of the learner's answer and the reference answer. This comparison must be improved because actually it is limited to nodes and edges, but does not include the layer attributes of the nodes (e.g.quantifiers). These attributes can be neglected in the reformulation of semantic networks (if networks without attributs are given), but not in free natural language answers. Additionally, we will implement several reference answers to one question and use the synonym list of our computer lexicon [1] to prevent misdetection of right answers. This point is again not critically for the reformulation since every correct answer is transformed to the reference semantic network, but of course, normally many different right answers exist to one question. While as far as we know our module for the analysis of semantic networks by logic inferences is unique there exist several approaches for the analysis of short answers. The most of them use predominantly statistically-based natural language processing and evaluation procedures (s. [5]). In contrast, our approach for free-text assessment based on a deep semantic understanding of the learner's answer. Thereby we hope that we can make a more detailed analysis of a text and can give a more detailed feedback to a learner.

References

[1] S. Hartrumpf, H. Helbig, and R. Osswald. The semantically based computer lexicon HaGenLex – Structure and technological environments. *Traitement automatique des langues*, volume 44(2), pages 81-105, 2003

[2] H. Helbig and C. Gnörlich. Multilayered extended semantic networks as a language for meaning representation in NLP systems. In: A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of LNCS, pages 69-85, Springer, 2002

[3] R. Lütticke. Problem Solving with Adaptive Feedback. In: P. De Bra and W. Nejdl, editors, Adaptive Hypermedia and Adaptive Wev-Based Systems – Third International Conference, AH 2004, volume 3137 of LNCS, pages 417-420, Springer, 2004

[4] R. Lütticke and H. Helbig. Practical courses in distance education supported by an interactive tutoring component. In: U. Benrath and A. Szücs, editors, *Supporting the Learner in Distance Education and E-Learning, Proceedings of the 3rd EDEN Research Workshop*, pages 441-447. Bibliotheks- und Informationssystem der Universität Oldenburg, 2004

[5] S. Valenti, F. Neri, and A. Cucchiarelli. An overview of current research on automated essay grading. Journal of Information Technology Education, volume 2, pages 319-330, 2003

[4] Since we are a German institute we teach only in this language and our research in NLP is focused on this language. Nevertheless we work on an English version of our parser and computer lexicon (HaGenLex [1]). The examples of the assessment in this paper are translated from the German.

PERO: a planning system for the explanation of problem solving in physics

Ali EL HORE [(2)]
Said TAZI [(1, 2)]

[(1)] *LAAS-CNRS*
7, avenue du Colonel Roche
31077 Toulouse Cedex 4

[(2)] *Université Toulouse I*
Place Anatole France
31042 Toulouse Cedex France

ali.elhore@univ-tlse1.fr, tazi@laas.fr

Abstract. The basic hypothesis of the ongoing research presented in this paper is that the observation of problem solving by students facilitates the learning process. We propose PERO a planner implemented in Java to solve exercises in physics automatically in such a way that the students can make observations on the proof. Several strategies for finding solutions can be chosen by the students and, an adequate explanation is generated. It is not only developed with the aim of solving exercises, but also serves for drafting answers and explanations in a language that combines formal and natural language. A complete application has been implemented based on exercises in electricity. The planning algorithm uses properties in physical science and mathematics such as axioms, laws, theorems, functions, etc. These properties are supplied by the teacher in a declarative way. A model based on the notion of intention has been adopted to help the generation of explanations. This model assumes that any action made by planer has its own intention. The intention is defined by the goal, the means and the reason for the action. The explanation of an exercise solution is a sequence of intentions of all actions performed by the planner to provide a proof for the exercise.

1. Introduction

In the college, the development of skills for resolving scientific problems must be considered as an essential component of fundamental learning. This ability should, therefore, be considered both as the target of personal development and as an important means of learning [1] and [2]. Teachers have a responsibility to find means of supporting this kind of learning process.

The basic hypothesis of the ongoing research presented in this paper is that the observation of problem solving by students facilitates the learning process. The externalisation of the problem solving process may help to facilitate this observation. The research presented here has for objective, on the one hand, to model knowledge involvement during the resolution of a problem and, on the other hand, to implement a planning algorithm for problem resolution, to explain to the learner the underlying mechanisms and to facilitate learning. The system proposed here is relatively general, even if it is presented here through its application in a physics course on electricity and, more precisely, on RLC circuits (Resistance, Impedance, Condenser). Indeed, PERO is a system conceived to adapt to various families of contents in the field of physical sciences. It is implemented as a prototype of problem solving in Java in a manner that makes it possible to calculate a complete solution to exercises in physics.

In this article, we present in the next section the planning process of Pero, in the third section a scenario describing through an example how the system solves a problem of LC circuits (impedance, condenser). A Graphical User Interface showing mixed information between explanation and the solution is also presented in the third section. The last section presents how the process of explanation is made.

2. The Pero Planner

The main algorithm combines all the small features and components of planning in a manner that is usable as a whole – this algorithm is inspired from [3] and [9]. Note that the input given to the planner combines the problem to solve and planning information (such as operators, initial and final states; the search tree is a typical component of problem solving). The first part of planning is based on substitutions and calculus, i.e. the planner proceeds by the transformations of terms from a state considered as an assumption state to a an other state called the successor state. If there is no solution, the system starts a set of function that transforms the terms of the assumption state trying to converge to the solution. The functions applied on terms are a set of mathematical possibilities such as derived function, Maxwell function, primitive function, and so on.
The following Figure 1 shows roughly the processes executed from the step in which the applet has obtained the user input until it prints out the solution.

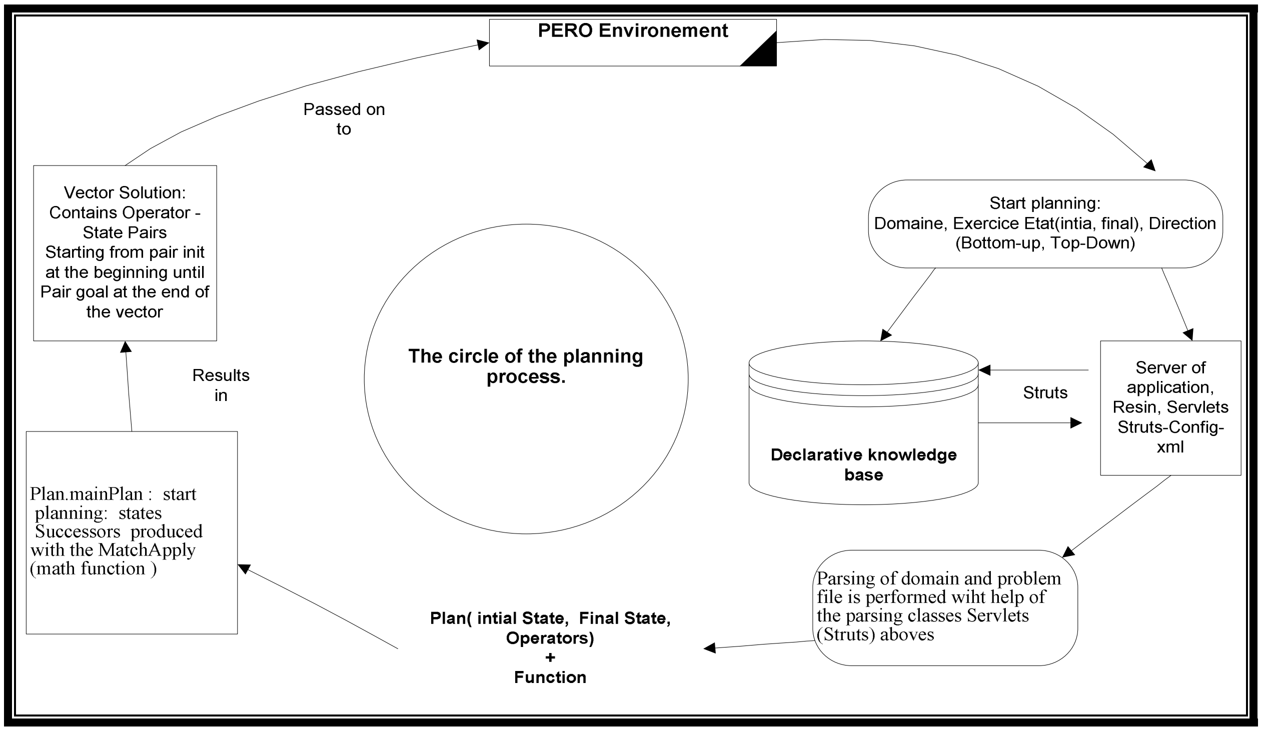

PERO Environement
Passed on to
Start planning: Domaine, Exercice Etat(intia, final), Direction (Bottom-up, Top-Down)
Vector Solution: Contains Operator - State Pairs Starting from pair init at the beginning until Pair goal at the end of the vector
The circle of the planning process.
Results in
Declarative knowledge base
Struts
Server of application, Resin, Servlets Struts-Config-xml
Plan.mainPlan : start planning: states Successors produced with the MatchApply (math function)
Parsing of domain and problem file is performed wiht help of the parsing classes Servlets (Struts) aboves
Plan(intial State, Final State, Operators) + Function

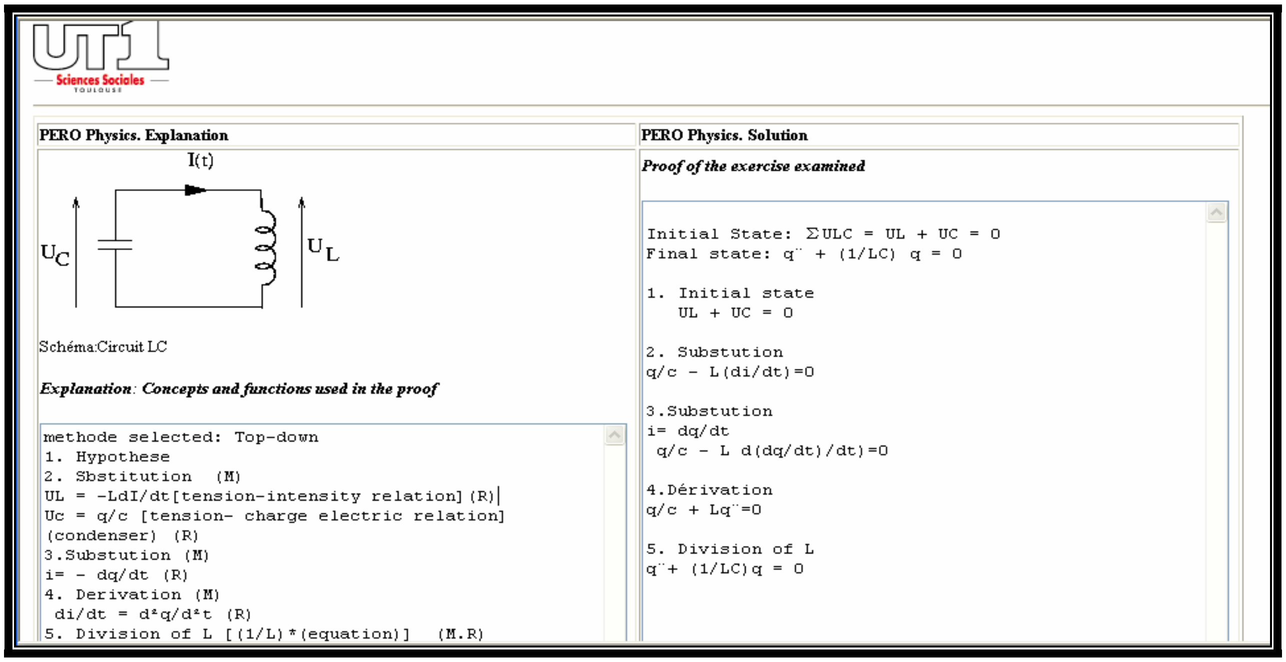
Figure 1: the planning process

3. *A* scenario *sample*

The knowledge is expressed declaratively in the system in the form of equalities. For example, the following equality expresses Ohm's law and the theorem of potential difference of a closed loop in the following form:
Ur = Ri [Ohm's Law] ;
$\sum U_{RLC} = U_R + U_L + U_C = 0$ *[The sum of the tensions of a closed loop is equal to zero]*

Problem solving in PERO can be compared to natural methods of deduction, when they are applied by a professor or a student. For example if the system must establish a differential equation of the electric charge (q) of a condenser of a circuit LC: $\ddot{q} + (1/LC)\, q = 0$.
The algorithm starts from the assumption state as initial state. In the example it is $\sum U_{LC} = 0$. The system uses the operators of substitution and calculation; it applies a set of transformations to all the terms of state assumption towards the following states until meeting a state in the which there is a conclusion. It will have built a graph of the proof. Explanatory texts associated with the equalities, with the operators and the functions used are concatenated to give the whole explanation of the solution. This explanation is expressed in terms of the intention model (Goal, Means, Reason) of each action. Figures 1 and 2 show the explanation process.
The use of a planning algorithm shows the incapacity of the Pero system to only? Solve certain questions while just being based on declaratory knowledge.

Given the quantity of knowledge to handle, it is of primary importance to store the data in an organized way. The planner will have to thus articulate it self and coordinates around a data base. A PHP MYSQL data base is used for this purpose.
In the proof process, one finds two methods applied to solving this exercise: the first top-down and the second one bottom-up. In the top-down method the planner builds the proof solution going from the initial state towards the final state. In the bottom-up method the planner builds the proof solution going from the final state towards the initial state.



PERO Physics. Explanation

Schéma:Circuit LC

Explanation*: *Concepts and functions used in the proof

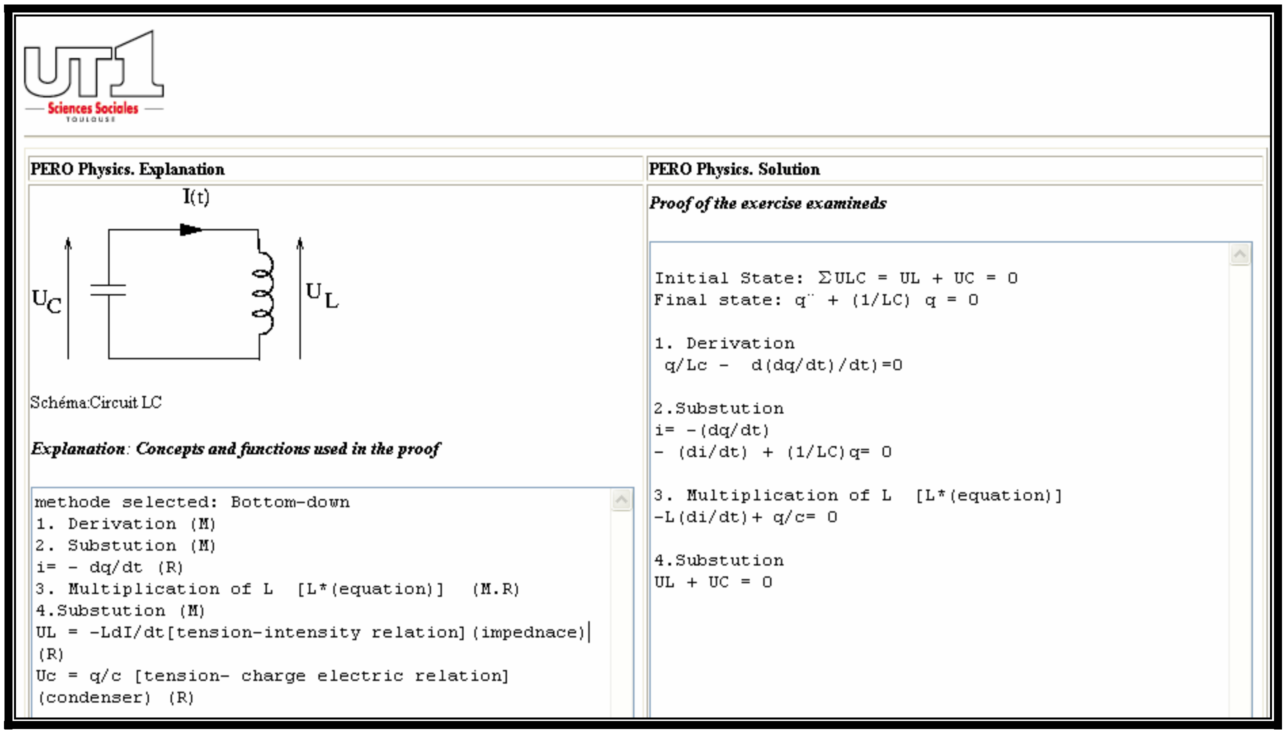
```
methode selected: Top-down
1. Hypothese
2. Sbstitution  (M)
UL = -LdI/dt[tension-intensity relation](R)
Uc = q/c [tension- charge electric relation]
(condenser) (R)
3.Substution (M)
i= - dq/dt (R)
4. Derivation (M)
 di/dt = d²q/d²t (R)
5. Division of L [(1/L)*(equation)]  (M.R)
```

PERO Physics. Solution

Proof of the exercise examined

```
Initial State: ΣULC = UL + UC = 0
Final state: q¨ + (1/LC) q = 0

1. Initial state
   UL + UC = 0

2. Substution
q/c - L(di/dt)=0

3.Substution
i= dq/dt
 q/c - L d(dq/dt)/dt)=0

4.Dérivation
q/c + Lq¨=0

5. Division of L
q¨+ (1/LC)q = 0
```

Figure 2: **graph of the proof of LC circuit with the use of the Top-down method.**



PERO Physics. Explanation

Schéma:Circuit LC

***Explanation**: Concepts and functions used in the proof*

```
methode selected: Bottom-down
1. Derivation (M)
2. Substution (M)
i= - dq/dt (R)
3. Multiplication of L  [L*(equation)]   (M.R)
4.Substution (M)
UL = -LdI/dt[tension-intensity relation](impednace)
(R)
Uc = q/c [tension- charge electric relation]
(condenser) (R)
```

PERO Physics. Solution

Proof of the exercise examineds

```
Initial State: ΣULC = UL + UC = 0
Final state: q¨ + (1/LC) q = 0

1. Derivation
 q/Lc -  d(dq/dt)/dt)=0

2.Substution
i= -(dq/dt)
- (di/dt) + (1/LC)q= 0

3. Multiplication of L  [L*(equation)]
-L(di/dt)+ q/c= 0

4.Substution
UL + UC = 0
```

Figure 3: **graph of the proof of circuit LC with the use of the method Bottom-up.**

4. Explanation

In the last three decades, Artificial Intelligence has developed automated systems for reasoning. In the eighties, most of this work was oriented towards systems that are able to give explanations about their own reasoning. The main result published from this research is that a good explanation is an explanation that takes account of the context. The model we propose here takes into account the context of actions being performed. Each action achieved by the planner is contextualized, i.e. we consider what one may call the intention of the action. The intention of an operator of the planner is a set of knowledge representing the goal of the action, the means used to perform the action and the reasons that justify the action. This knowledge depends on the context of the action, so for any action performed to solve a problem, there is an intention that could be considered as the explanation of this contextual action. The whole explanation of the solution is considered as the set of explanations of the actions performed to attain the final solution. In previous work [10] we have developed the model of Intentional structures that we recall briefly here.

Pero generates the knowledge concerning the description of what we call the intention of the action, (i.e. the goal, the means and the reasons for the action). This knowledge comes from the solution graph. The whole explanation destined for the student is the concatenation of the all intentions of the actions belonging to the path solution.

In order to illustrate this model, the following is a draft of how the solution is proved and the explanation is generated.

Let EQ1 be the initial state, and EQ2 be the final state. (EQ1 and EQ2 are respectively the first equations that will lead to the second equation after a certain number of substitutions and or calculus).
When the system passes from one state S1 to the following one (S2) it concatenates the intention of the action that leads from S1 to S2.
For the global problem, the intention of the problem is defined by:
1. Goal: try all possible combination of substitution and calculus to find the solution
2. Means: are the operators used in the actions;
3. Reason: is the set of theorems, laws or functions that triggers the operator.

For each action the intention is defined as

1. Goal: Try to find the final state from the current state
2. Means: The operators used to perform the action (e.g. Substitute, Calculate, Derive, etc.)
3. Reason: justify the action by the arguments that trigger the action, these arguments can be theorems, laws, lemma functions, etc.)

The proof paths of the LC circuit, in Figures 1 and 2 show how the explanation is presented to the student (B for the Goal (B as in the word '*But*' in French), M for the Means and R for the reason) in both top-down and bottom-up strategies. In these figures one can see the explanation of the adopted solution.

5. Conclusion

This work aims to support learning by giving students the possibility of observing how a problem can be resolved.
We propose PERO a planner implemented in Java to solve exercises in physics automatically in a way that allows the students to make observations on the proof. Several strategies (such as top-down and bottom-up strategies) for finding solutions can be chosen by the students and, an adequate explanation is generated. It is not only developed with the aim of solving exercises, but also for drafting answers and explanations in a language that combines formal and natural language. The explanation is generated from the solution graph in which knowledge about the goal, the means and the reason for each action is stored.
The next step of this ongoing work is to validate this system with a group of users.

As a possible extension to this work, we plan to add a module for the generation of a sheet of exercises using automatic indexing of solutions already stored in the system. The indexation will be made on concepts such as theorems of physical sciences used for their resolution. This kind of indexation applied to mathematical exercises has been developed by [7].

Bibliography

[1] Lise Poirier Prou, Enseigner et apprendre la résolution de problèmes, tiré du volume 11, no 1, de la revue Pédagogie collégiale, paru en octobre 1997 (p. 18-22).

[2] TARDIF, J. (1992), Pour un enseignement stratégique. L'apport de la psychologie cognitive, Montréal, Les Éditions Logiques.

[3] Schmid, U. (2003). “Inductive Synthesis of Functional Programs Learning Domain-Specific Control Rules and Abstract Schemes”. Springer, LNAI 2654. Also available at: http://www.inf.uos.de/schmid/pub-ps/habil.pdf (16.11.2003).

[4] Russell, S. J., Norvig P. (1995). “Artificial intelligence: A Modern Approach”. Englewood Cliffs, NJ: Prentice Hall.

[5] Malik Ghallab, Planification et décision, To appear in : La Robotique Mobile, J.P. Laumaond (ed.), Hermes, 2001

[6] Kristine S. Lundi, Analyse de l’activité explicative en interaction, ‘Etude de dialogues d’enseignants de physique en formation interprétant les interactions entre èlèves’. Thèse soutenu en 27 octobre 2003.

[7] Jean-Marc Labat et all. (2003) ‘ Génération de feuille d’exercices de géométrie à l’aide d’énoncés indexés automatiquement’, Environnements Informatique pour l’Apprentissage Humain, Strasbourg 2003.

[8] Natalya F. Noy et Deborah L. McGuiness. ‘ Développement d’une ontologie 101 : Guide pour la création de votre première ontologie’. Bureau de normalisation documentaire.

[9] E. Aktolga. 2004 ‘A Java planner for Blocksworld problems’. University of Osnabrueck

[10] Tazi, S., and Evrard, F.. Intentional structures of documents, ACM Hypertext 01, Århus, Denmark. August 2001.