Discourse Processing for Explanatory Essays in Tutorial Applications

Pamela W. Jordan and Kurt VanLehn
Learning Research and Development Center
University of Pittsburgh
Pittsburgh PA 15260
[pjordan,vanlehn]@pitt.edu

Abstract

The Why-Atlas tutoring system presents students with qualitative physics questions and encourages them to explain their answers via natural language. Although there are inexpensive techniques for analyzing explanations, we claim that better understanding is necessary for use within tutoring systems. In this paper we describe how Why-Atlas creates and utilizes a proof-based representation of student essays. We describe how it creates the proof given the output of sentence-level understanding, how it uses the proofs to give students feedback, some preliminary runtime measures, and the work we are currently doing to derive additional benefits from a proof-based approach for tutoring applications.

1 Introduction

Whereas most explanations are produced and adapted to benefit or inform a hearer, a self-explanation is produced for the benefit of the speaker. If there is a hearer he often already knows all about the topic as in a tutoring context. Self-explanation is a cognitively valuable pedagogical activity because it leads students to construct knowledge (Chi et al., 1994), and it can expose deep misconceptions (Slotta et al., 1995). But it is difficult to encourage self-explanation without giving the student substantive feedback on what they generate (Aleven and Koedinger, 2000; Chi et al., 2001). To give substantive feedback the system has to be able to understand student explanations to some degree.

The Why-Atlas system presents students with qualitative physics problems and encourage them to write their answers along with detailed explanations for their answers. While physics misconceptions have proven to be particularly resistant to repair, practice with qualitative physics questions helps in overcoming some of these misconceptions (Hake, 1998).

The student explanation shown in (1), which is from our corpus of human-human computer-mediated tutoring sessions, illustrates how challenging these explanations are for a system to understand. The problems we have examined require a short essay with an average of 6.9 sentences to fully explain to the satisfaction of experienced physics instructors.

(1) Question: Suppose you are running in a straight line at constant speed. You throw a pumpkin straight up. Where will it land? Explain.

Explanation: Once the pumpkin leaves my hand, the horizontal force that I am exerting on it no longer exists, only a vertical force (caused by my throwing it). As it reaches it's maximum height, gravity (exerted vertically downward) will cause the pumpkin to fall. Since no horizontal force acted on the pumpkin from the time it left my hand, it will fall at the same place where it left my hands.

Statistical text classification approaches, such as latent semantic analysis (Landauer et al., 1998), have shown promise for classifying a student explanation into medium-grained good and bad categories (Graesser et al., 2000). For instance, a medium-

grained category that should match (1) is the often-observed impetus misconception:

> If there is no force on a moving object, it slows down.

Such medium-grained categories typically have multiple propositions and contain multiple content words. While successful with medium-grained classes, statistical approaches are not yet able to distinguish subtle but important differences between good and bad explanations. Statistical classification is insensitive to negations[1], anaphoric references[2], and argument ordering variations[3] and its inferencing is weak[4]. To capture these subtle differences and to allow us to respond more directly to what the student actually said[5], we need the precision possible so far only with symbolic approaches. So Why-Atlas parses each sentence into a propositional representation.

The PACT Geometry Tutor is an operational prototype that does a finer-grained symbolic classification (Aleven et al., 2001). PACT also parses a student explanation into a propositional representation but then uses LOOM to classify these into fine-grained categories that typically express one proposition. This approach looks promising (Aleven et al., 2001), but the system's goal is to elicit a justification for a step in a geometry proof and generally these can be expressed with a single sentence that succinctly translates into a small number of propositions. It isn't clear that this approach will work well for the longer, more complex explanations that the Why-Atlas system elicits.

Instead of classifying propositions, the Why-Atlas system constructs abductive proofs of them. A proof-based approach gives more insight into the line of reasoning the student may be following across multiple sentences because proofs of the propositions share subproofs. Indeed, one proposition's entire proof may be a subproof of the next proposition. Moreover, subtle misconceptions such as impetus are revealed when they must be used to prove a proposition.

Abductive inference has a long history in plan recognition, text understanding and discourse processing (Appelt and Pollack, 1992; Charniak, 1986; Hobbs et al., 1993; McRoy and Hirst, 1995; Lascarides and Asher, 1991; Rayner and Alshawi, 1992). We are using an extended version of SRI's Tacitus-lite weighted abductive inference engine (Hobbs et al., 1993) as our main tool for building abductive proofs. We had to extend it in order to use it for domain as well as language reasoning. As advised in (Appelt and Pollack, 1992), abductive inference requires some application specific engineering to become a practical technique.

In this paper we describe how the system creates and utilizes a proof-based representation of student essays. We describe how it creates the proof given the output of sentence-level understanding, how it uses the proofs to give students feedback, some preliminary run-time measures, and the work we are currently doing to derive additional benefits from a proof-based approach for tutoring applications.

First we give an overview of the Why-Atlas tutoring system architecture. Next we give some background on weighted abduction and Tacitus-lite+ and describe how it builds an abductive proof. Next we describe how the system uses the proofs to give students feedback on their essays. Finally, we discuss efficiency issues and our future evaluation plans.

2 Overview of the Why-Atlas Tutoring System

The architecture for the Why-Atlas qualitative physics tutoring system is shown in Figure 1. The user interface for the system is a screen area in which the physics question is displayed along with an essay entry window and a dialogue window. As the student enters an answer and explanation for a qualitative physics question the sentence-level understanding module builds sets of propositions and passes

[1] A good explanation followed by "But I don't think that will happen." would be classified as good.

[2] In (1) above, it would tend to misclassify the last clause as the correct answer "the pumpkin will land in my hands" because it does not understand the temporal anaphora.

[3] The difference between x accelerates faster than y and y accelerates faster than x would not be detected.

[4] In (1), the student has the extreme belief that the pumpkin has **no** horizontal velocity. This would probably not be recognized as a case of "slowing down" by statistical classification.

[5] When a true statement lacks precision, the tutor should acknowledge the correct statement and elicit more precision rather than continuing as if it were wrong. For example, if a student makes a correct statement about the velocity of an object but did not report it in terms of the horizontal and vertical components of the velocity, the tutor should ask which was intended.

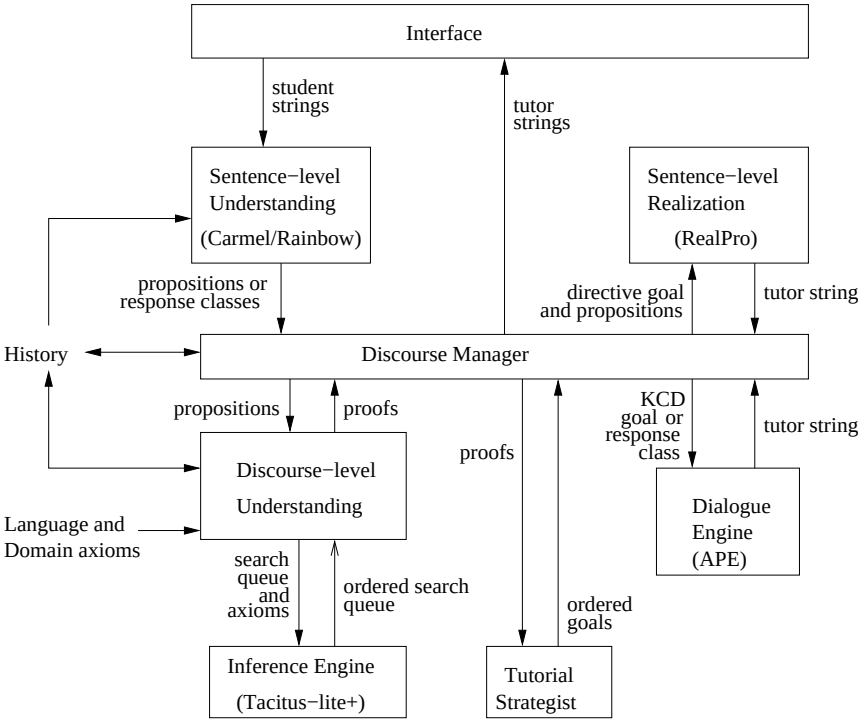

Figure 1: Why-Atlas Tutoring System Architecture

them, via the discourse manager, to the discourse-level understanding module. Each set of propositions represents one interpretation of a sentence. The user interface and the sentence-level understanding components are described in detail in (Rosé, 2000; Freedman et al., 2000).

The discourse-level understanding module uses language and domain reasoning axioms and the Tacitus-lite+ abductive inference engine to create a set of proofs that offer an explanation for the student's essay and give some insight into what the student may believe about physics and how to apply that knowledge. The discourse-level understanding module updates the propositions and the search queue for proofs in the history with the results from Tacitus-lite+. This part of the history supports anaphora resolution and processing of revisions a student may make to his essay. The discourse manager module selects and sends the *best* proofs to the tutorial strategist.

The tutorial strategist identifies relevant communicative goals. Currently there are four categories of communicative goals. Two of these, disambiguating terminology and clarifying the essay, are addressed via directives to modify the essay. The other two, remediating misconceptions and eliciting more complete explanations, are addressed via dialogue. Misconceptions are detected when the proof includes an axiom that is incorrect or inapplicable. Incompleteness is detected under two conditions. First, there may be multiple proofs that are equally good. This condition indicates that the student did not say enough in his explanation for the system to decide which proof best represents what the student's reasoning may be. Each possible line of reasoning could point to different underlying problems with the student's physics knowledge. The second condition occurs when the student fails to explicitly state a *mandatory point*, which is a proposition that domain instructors require of any acceptably complete essay. Once the tutorial strategist has identified communicative goals it prioritizes them according to curriculum constraints and sends them to the discourse manager, which selects the highest priority goal after taking dialogue coherency into account and sends the goal to either the dialogue engine or the sentence-level realization module.

The dialogue engine initiates and carries out a dia-

logue plan that will either help the student recognize and repair a misconception or elicit a more complete explanation from the student. The main mechanism for addressing these goals are what we call a knowledge construction dialogue (KCD) specification. A KCD specification is a hand-authored pushdown network. Nodes in the network are either the system's questions to students or pushes and pops to other networks. The links exiting a node correspond to anticipated responses to the question. Each question is a canned string, ready for presentation to a student. The last state of the network is saved in the history and the sentence-level understanding module accesses this in order to get information for analysis of student responses. The sentence-level understanding module uses a classification approach for dialogue responses from the student since currently the dialogue plans are limited to ones that expect short, direct responses. During a dialogue, response class information is delivered directly to the dialogue engine via the discourse manager. The dialogue engine is described further in (Rosé et al., 2001).

The other communicative goals, disambiguating terminology and clarifying the essay, are addressed by the discourse manager as directives for the student to modify the essay. It passes propositions and a goal to the sentence-level realization module which uses templates to build the deep syntactic structures required by the RealPro realizer (Lavoie and Rambow, 1997) for generating a string that communicates the goal.

When the discourse manager is ready to end its turn in the dialogue, it passes the accumulated natural language strings to the user interface. This output may also include transitions between the goals selected for the turn.

While a dialogue is in progress, the discourse-level understanding and tutorial strategist modules are bypassed until the essay is revised. Once the student revises his essay, it is reanalyzed and the cycle repeats until no additional communicative goals arise from the system's analysis of the essay.

Although the overall architecture of the system is a pipeline, there is feedback to earlier modules via the history. Only the discourse-level understanding and discourse manager modules are internally pipelines, the rest are rule-based.

3 Background on Weighted Abduction and Tacitus-lite+

Abduction is a process of reasoning from an observation to possible explanations for that observation. In the case of the Why-Atlas system the observations are what the student said and the possible explanations for why the student said this are the physics qualitative axioms (both good and bad) and orderings of those axioms that support what the student said. To arrive at the explanation, some assumptions have to be made along the way since all the inferences that underly an explanation will not be expressed.

Weighted abduction is one of several possible formalisms for realizing abductive reasoning. With weighted abduction there is a cost associated with making an assumption during the inference process. Following the weighted abductive inference algorithm described in (Stickel, 1988), Tacitus-lite is a collection of axioms where each axiom is expressed as a Horn clause. Further, each conjunct p_i has a weight w_i associated with it, as in (2). The weight is used to calculate the cost of assuming p_i instead of proving it where $cost(p_i) = cost(r) * w_i$.

(2) $p_1 w_1 \wedge \cdots \wedge p_n w_n \Rightarrow r$

Given a goal or observation to be proven, Tacitus-lite takes one of four actions; 1) assumes the observation at the cost associated with it 2) unifies with a fact for zero cost 3) unifies with a literal that has already been assumed or proven at no additional cost 4) attempts to prove it with an axiom.

All possible proofs could be generated. However, Tacitus-lite allows the applications builder to set depth bounds on the number of axioms applied in proving an observation and on the global number of proofs generated during search. Tacitus-lite maintains a queue of proofs where the initial proof reflects assuming all the observations and each of the four above actions adds a new proof to the queue. The proof generation can be stopped at any point and the proofs with the lowest cost can be selected as the most plausible proofs for the observations.

Tacitus-lite uses a best-first search guided by heuristics that select which proof to expand, which observation or goal in that proof to act upon, which action to apply and which axiom to use when that is

the selected action. Most of the heuristics in Why-Atlas are specific to the domain and application.

SRI's release of Tacitus-lite was subsequently extended by the first author of this paper for the research project described in (Thomason et al., 1996). It was named Tacitus-lite+ at that time. Two main extensions from that work that we are making use of are: 1) proofs falling below a user defined cost threshold halt the search 2) a simple variable typing system reduces the number of axioms written and the size of the search space (Hobbs et al., 1988, pg 102).

Unlike the earlier applications of Tacitus-lite+, Why-Atlas uses it for both shallow qualitative physics reasoning and discourse-level language reasoning. To support qualitative physics reasoning we've made a number of general inference engine extensions, such as improved consistency checking, detecting and avoiding reasoning loops and allowing the axiom author to express both good and bad axioms in the same axiom set. These recent extensions are described further in (Jordan et al., 2002).

4 Building an Abductive Proof

The discourse-level understanding module uses language axioms and the Tacitus-lite+ abductive inference engine to resolve pronominal and temporal anaphora and make other discourse-level language related inferences. It transforms the sentence-level propositions into more complete propositions given the context of the problem the student is solving (represented as facts) and the context of the preceding sentences of the essay.

From these discourse-level propositions, proofs are built and analyzed to determine appropriate communicative actions. To build these proofs, the discourse-level understanding module uses domain axioms, the above resulting propositions and again the Tacitus-lite+ abductive inference engine.

We've separated the discourse-level language axioms from the domain axioms both for efficiency and modularity because there is generally only a small amount of interaction between the language and domain axioms. Separating them reduces the search space. In cases where interaction within a single axiom is necessary, we've place these axioms in the set of language axioms. The system currently has 90 language axioms and 95 domain axioms. The domain axioms fully cover 5 problems as well as parts of many other problems.

We will describe in more detail each of these stages of building the proof in the sections that follow.

4.1 Applying Discourse-level Language Axioms to Sentence-level Propositions

The discourse-level language axioms are currently addressing the local resolution of pronominal and temporal anaphora, flattening out embedded relationships and canonicalizing some lexical choices that can only be resolved given the context of the problem. We are still developing and testing axioms that will better address pronominal and temporal anaphora inter-sententially and axioms that will generate additional propositions for quantifiers and plurals.

Pronominal Anaphora. It is generally easy to resolve pronominal anaphora in the context of a qualitative physics problem because there are a small number of candidates to consider. For example, in the case of the pumpkin problem in (1), there are only four physics bodies that are likely to be discussed in a student essay; the pumpkin, the runner, the earth and air.

The system is able to resolve simple intra-sentential pronominal references using language axioms. The objects described in a single sentence are the candidate set and argument restrictions rule out many of these candidates. But to resolve inter-sentential anaphora, as in (3), the system currently relies on the domain axioms. The domain axioms will bind the body variables to their most likely referents during unification with facts, and previously assumed and proven propositions similarly to (Hobbs et al., 1988).

(3) The man is exerting a force on *it*.

But in the case of anaphoric references to physical quantities such as velocity, acceleration and force, as in (4), we need to extend the language axioms to handle these cases because it involves too much unconstrained search for the domain axioms to resolve these. This is because the physical quantities are the

predicates that most strongly influence the domain reasoning.

(4) *The velocity* is constant before the pumpkin is thrown. But after the release, *it* will decrease because there is no force.

To extend the language axioms to address inter-sentential anaphora we need to implement and test a recency ordering of the physics bodies and quantities that have already been discussed in the essay. But we expect this to be simple to do since the essays generally only involve one discourse segment.

Temporal Anaphora. As with pronominal anaphora, temporal anaphora is usually clear because the student often explicitly indicates when an event or state occurs relative to another event or state as with the first sentence of the explanation presented in (1). In these cases, the domain-level reasoning will be able to unify the anchor event or state with an already known event or state in the proof it is constructing.

When there is no temporal anchor the domain-level search is too under-constrained so the language axioms resolve the temporal orderings. In some cases world knowledge is used to infer the temporal relationships as in (5). Here we know that to catch an object it must have been thrown or dropped beforehand and so the event in (5a) must occur after the event in (5b).

(5) a. The man catches the pumpkin.

b. This is because they had the same velocity when he threw it.

Otherwise, the language axioms use information about tense and aspect and default orderings relative to these to guide inferences about temporal relationships ((Kamp, 1993; Dowty, 1986; Partee, 1984; Webber, 1988) inter alia).

Embedded Relationships. In the physics essays we are addressing, there is a tendency to express multiple relations within a single sentence as in (6). Here the "equal" and "opposite" relations are embedded in a temporal "when" relation. In this case the sentence-level understanding module is not in the best position to indicate the specific constraints that each of these relations imposes so this is handled by discourse-level understanding. It would also impose a greater burden on the domain-level proof building if these relationships were not resolved beforehand. For example, in the case of the last clause in (6) there is an elliptical reference that could cause the domain-level a great deal of unconstrained search.

(6) When the magnitude of the pumpkin's velocity equals the man's, the pumpkin's velocity is in the opposite direction.

Canonicalizing Lexical Usage. One simple case in which the language axioms canonicalize lexical items has to do with direction. For example, saying "move up the inclined plane" should be interpreted as a positive direction for the horizontal component even though the phrase contains "up". The axioms are able to canonicalize references such as up, down, left, right, north, south into a positive or negative direction relative to an axis in a coordinate system that may be tilted slightly to align with planes. This is an example of the kinds of axioms in which language and domain knowledge are interacting within a single axiom.

Quantifiers and Plurals In our target essays, there is frequent usage of quantifiers and plurals with respect to physics bodies and frequent use of quantifiers with respect to parameters of physical quantities (e.g. "at all times" "all the magnitudes of the velocities").

We have recently completed our specification for a sentence-level representation of quantifiers and plurals. From this representation the language axioms will generate an appropriate number of new propositions to use in the proof building stage, given the context of the problem and the expression recognized from sentence-level processing.

Although we have not yet implemented and tested this set of language axioms, we have successfully hand-encoded sentences such as (7) into both their sentence-level and discourse-level representations and have used the latter successfully in the final proof building process. For example, for (7), the system creates two equivalent propositions about acceleration, each referring to different balls. In addition, both of these propositions are related to two

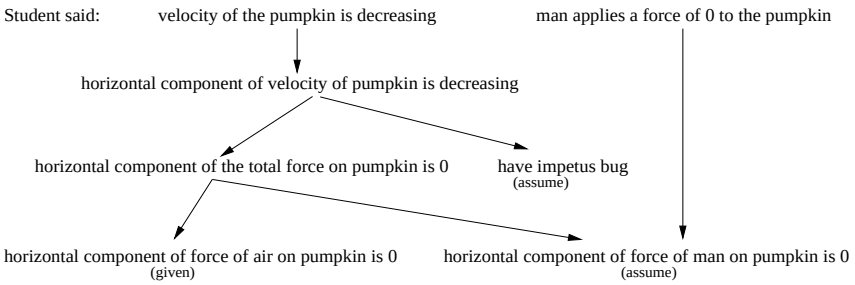

Figure 2: Example of Simplified Abductive Proof for "The pumpkin moves slower because the man is not exerting a force on it."

additional propositions about the force of gravity applying to the same ball as in its related acceleration proposition.

(7) The acceleration of both balls is increasing due to the force of earth's gravity.

4.2 Applying Domain-level Axioms to Build an Explanatory Proof

The propositions produced by applying the language axioms are the goals that are to be proven using domain-level axioms. Figure 2 is an example of a simplified abductive proof for sentence (8).

(8) The pumpkin moves slower because the man is not exerting a force on it.

Each level of downward arrows from the gloss of a proposition in Figure 2 represents a domain axiom that can be used to prove that proposition. One way to prove that the velocity of the pumpkin is decreasing is to prove that just the horizontal component of the velocity vector is the one that is decreasing since the context of the question (see (1)) makes this a likely interpretation. Alternatively, the system could request that the student be more precise by asking which components of the velocity vector are decreasing.

In the case of trying to prove that the horizontal component is decreasing, Tacitus-lite+ is applying a bad physics axiom that is one manifestation of the impetus misconception; the student thinks that a force is necessary to maintain a constant velocity. In this case it assumes the student has this misconception but alternatively the system could try to gather more evidence that this is true by asking the student diagnostic questions.

Next Tacitus-lite+ proves that the total force on the pumpkin is zero by proving that the possible addend forces are zero. In the context of this problem, it is a given that air resistance is negligible and so it unifies with a fact for zero cost. Next it assumes that the student believes the man is applying a horizontal force of 0 to the pumpkin.

Finally, it still needs to prove another proposition that was explicitly asserted by the student; that the force of the man on the pumpkin is 0. As with the velocity, it will try to prove this by proving that the horizontal component of that force is zero. Since it has already assumed that this is true, the abductive proof is finished and ready to be further analyzed by the tutorial strategist module to give additional feedback to the student.

4.3 Incrementally Processing an Essay

We have also extended Tacitus-lite+ to run incrementally so that it can start processing before the student completes his essay. In this way it can take advantage of the processing lull as the student composes his essay. In simulations of various typing speeds, (Rosé et al., 2002) estimated that there is a 60 second processing lull during the completion of a sentence after subtracting out a 5 second average incremental parsing cost. During this lull it can build proofs using the previous sentences in the essay.

To run Tacitus-lite+ incrementally, we added a function that takes as input a proof queue and the new goals that are to be proven and returns a new proof queue. The discourse-level understanding module builds the input proof queue by finding the proofs in the most recent queue with which the new goals are consistent and adding the new goals to a copy of each of those proofs. We then modified

Tacitus-lite+ to take an arbitrary proof queue as input.

The discourse-level understanding module stores and selects proof queues, which are returned by Tacitus-lite+ after it attempts to prove a sentence. Suppose for example that each sentential input is treated as a separate input to Tacitus-lite+ and that sentence S_k has already been processed and yielded proof queue Q_k. As the next sentence S_{k+1} arrives, a copy of Q_k is updated with proofs that include S_{k+1} as new information to be proven. But if S_{k+1} conflicts with every proof in the copy of Q_k, then an earlier proof queue is tried. Similarly, if a student modifies a previously processed sentence, the original sentence is regarded as having been deleted. The inference process backs up to the point just before the deleted sentence was processed and reprocesses the substituted sentence and all that follows it. This mechanism for backing-up allows the inference process to be incremental.

At the end of composing an essay, the student will in the best case have to wait the length of time that it takes to finish parsing the last sentence of the essay plus the length of time that it takes to extend the proof by one sentence. In the worst case, which is when he modifies the first sentence or inserts a new first sentence, he will have to wait the same amount of time as he would for non-incremental discourse-level understanding.

5 Deriving Feedback for Students From Plausible Proofs

To identify communicative goals the tutorial strategist next analyzes the best proofs. Currently it examines just one of the best proofs by applying a set of test patterns to parts of the proof. It can test for combinations of patterns for givens (mainly to get bindings for variables in a pattern), for assumed propositions, for propositions asserted in the student's essay, and for inferred propositions. In addition it can also test for missing patterns in the proof and for particular domain axioms to have been used. Each goal that the system is capable of addressing is linked to sets of patterns that are expected to be indicative of it. In the case of the proof for (8), the tutorial strategist identifies a dialogue goal that addresses the impetus misconception as being relevant since an impetus axiom is part of the proof.

In addition to engaging students in a dialogue, the system can also give direct, constructive feedback on the essays they are composing. When there are multiple interpretations, it is better to ask the student to make certain things in the essay clearer. The tutorial strategist includes test patterns that target important details that students often leave out. For example, suppose the student says that the velocity is increasing but this is only true for the vertical component of the velocity vector. It may then be important to clarify which component of the velocity the student has in mind since thinking that the horizontal component is increasing indicates a misconception.

It is also possible that two propositions in an essay will be contradictory. In this case the system points out that there is a conflict, describes the conflict and directs the student to repair it.

We expect to extend the tutorial strategist module so that if there are multiple best proofs, it will ask the student questions that will help it disambiguate which proof is most representative of the student's intended meaning for the essay.

6 Preliminary Results and Future Plans

Although we've found that incremental understanding is successful at taking advantage of the processing lull during which the student composes his essay, we still need to fine-tune it so as to minimize both the need to back-up and how much underconstrained searching it does (i.e. the more Tacitus-lite+ has of the student's explanation the more constrained the search is). Currently, Tacitus-lite+ runs after every new sentence that is recognized by the sentence-level understanding module. During each of these runs Tacitus-lite+ continues until one of its run-time thresholds is exceeded.

We plan to also experiment with other ways of bounding the run-time for Tacitus-lite+ during incremental processing. For example, we might impose a specific time-limit that is based on the expected 60 second processing lull while the student composes his next sentence.

In initial timing tests, using a set of 5 correct essays that involved no backing up, the average incremental processing time per sentence when we set the search bound to 50 proofs and the assumption cost

threshold to .05[6], is 21.22 seconds. The worst case time for extending a proof by one sentence was 98 seconds and the best was 1 second. So in the best case, which is when no previous sentences have been modified, the student will wait on average 21.22 seconds after he completes the last sentence in his essay for a response from Why-Atlas.

In human-human computer-mediated tutoring, we found that in the worst case the student waits 2 minutes for a reply from the tutor after completing the essay. The wait time in the case of the human tutor is a combination of the time it takes to read and analyze the student's response and then compose a reply.[7] Although the timings are inconclusive and not directly comparable, it gives us an order of magnitude for tolerable wait times.

We will complete a 5 week formative evaluation of the Why-Atlas system in which we will compare the learning gains of 24 students to other sets of students in three other conditions; 1) a text control 2) human tutoring 3) another tutoring system that uses statistical classification only. During these trials, we will log decisions and processing times for each module of the system. From these detailed logs we will be able to better evaluate the speed and correctness of each system module.

Acknowledgments

This research was supported by MURI grant N00014-00-1-0600 from ONR Cognitive Science and by NSF grant 9720359. We thank the entire NLT team for their many contributions in creating and building the Why-Atlas system. In particular we thank Michael Ringenberg, Maxim Makatchev, Uma Pappswamy and Michael Böttner for their work with Tacitus-lite+ and the domain axioms and Roy Wilson for his work with the sentence-level realization module.

[6] An assumption cost of 1 means everything is assumed and a cost of 0 means that nothing was assumed.

[7] In these timing studies, we also did not allow the tutor to see the student input until the student had finished composing it. This was because our previous experiences with computer-mediated human tutoring have shown that some human tutors have a propensity for referring to something the student had started to write and then deleted. Our goal was to try to collect interactions that would be closer to those we expected with an intelligent tutoring system and was not primarily for comparing efficiency of a computer tutor to a human one.

References

Vincent Aleven and Kenneth R. Koedinger. 2000. The need for tutorial dialog to support self-explanation. In *Building Dialogue System for Tutorial Applications, Papers of the 2000 AAAI Fall Symposium*.

Vincent Aleven, Octav Popescu, and Kenneth R. Koedinger. 2001. A tutorial dialogue system with knowledge-based understanding and classification of student explanations. In *Working Notes of 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Douglas Appelt and Martha Pollack. 1992. Weighted abduction for plan ascription. *User Modeling and User-Adapted Interaction*, 2(1 – 2):1 – 25.

Eugene Charniak. 1986. A neat theory of marker passing. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI'86)*, pages 584 – 588.

Michelene T. H. Chi, Nicholas de Leeuw, Mei-Hung Chiu, and Christian LaVancher. 1994. Eliciting self-explanations improves understanding. *Cognitive Science*, 18:439–477.

Michelene T. H. Chi, Stephanie A. Siler, Heisawn Jeong, Takashi Yamauchi, and Robert G. Hausmann. 2001. Learning from human tutoring. *Cognitive Science*, 25(4):471–533.

David Dowty. 1986. The effects of aspectual class on the temporal structure of discourse: Semantics or pragmatics? *Linguistics and Philosophy*, 9(1).

Reva Freedman, Carolyn Ros´e, Michael Ringenberg, and Kurt VanLehn. 2000. ITS tools for natural language dialogue: A domain-independent parser and planner. In *Proceedings of the Intelligent Tutoring Systems Conference*.

Arthur C. Graesser, Peter Wiemer-Hastings, Katja Wiemer-Hastings, Derek Harter, Natalie Person, and the TRG. 2000. Using latent semantic analysis to evaluate the contributions of students in autotutor. *Interactive Learning Environments*, 8:129–148.

Richard R. Hake. 1998. Interactive-engagement versus traditional methods: A six-thousand student survey of mechanics test data for introductory physics students. *American Journal of Physics*, 66(4):64–74.

Jerry Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. 1988. Interpretation as abduction. In *Proc. 26th Annual Meeting of the ACL, Association of Computational Linguistics*, pages 95–103.

Jerry Hobbs, Mark Stickel, Douglas Appelt, and Paul Martin. 1993. Interpretation as abduction. *Artificial Intelligence*, 63(1–2):69–142.

Pamela W. Jordan, Maxim Makatchev, Michael Ringenberg, and Kurt VanLehn. 2002. Engineering the Tacitus-lite weighted abductive inference engine for use in the Why-Atlas qualitative physics tutoring system. Manuscript, University of Pittsburgh.

Hans Kamp. 1993. *From Discourse to Logic; Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht Holland.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284.

Alex Lascarides and Nicholas Asher. 1991. Discourse relations and defeasible knowledge. In *29th Annual Meeting of the Association for Computational Linguistics*, pages 55 – 62.

Benoit Lavoie and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing Chapter of the Association for Computational Linguistics*, pages 265–268, Washington, D.C.

Susan McRoy and Graeme Hirst. 1995. The repair of speech act misunderstandings by abductive inference. *Computational Linguistics*, 21(4):435–478, December.

Barbara Partee. 1984. Nominal and temporal anaphora. *Linguistics and Philosophy*, 7:243 – 286.

Manny Rayner and Hiyan Alshawi. 1992. Deriving database queries from logical forms by abductive definition expansion. In *Proceedings of the Third Conference of Applied Natural Language Processing*, pages 1 – 8, Trento, Italy.

Carolyn Ros´e, Pamela Jordan, Michael Ringenberg, Stephanie Siler, Kurt VanLehn, and Anders Weinstein. 2001. Interactive conceptual tutoring in atlas-andes. In *Proceedings of AI in Education 2001 Conference*.

Carolyn P. Ros´e, Antonio Roque, Dumisizwe Bhembe, and Kurt VanLehn. 2002. An efficient incremental architecture for robust interpretation. In *Proceedings of Human Language Technology Conference*, San Diego, CA.

Carolyn P. Ros´e. 2000. A framework for robust semantic interpretation. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*.

James D. Slotta, Michelene T.H. Chi, and Elana Joram. 1995. Assessing students' misclassifications of physics concepts: An ontological basis for conceptual change. *Cognition and Instruction*, 13(3):373–400.

Mark Stickel. 1988. A prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. Technical Report 451, SRI International, 333 Ravenswood Ave., Menlo Park, California.

Richmond H. Thomason, Jerry Hobbs, and Johanna D. Moore. 1996. Communicative goals. In K. Jokinen, M. Maybury, M. Zock, and I. Zukerman, editors, *Proceedings of the ECAI 96 Workshop Gaps and Bridges: New Directions in Planning and Natural Language Generation*.

Bonnie Webber. 1988. Tense as discourse anaphor. *Computational Linguistics*, 14(2):61 – 71.