# Initiative in Tutorial Dialogue Systems

**Greg A. Keim, Michael S. Fulkerson and Alan W. Biermann**
Department of Computer Science
Duke University
{keim,msf,awb}@cs.duke.edu

**Abstract**

Tutorial dialogue offers several interesting challenges to mixed-initiative dialogue systems. In this paper, we outline some distinctions between tutorial dialogues and the more familiar task-oriented dialogues, and how these differences might impact our ideas of focus and initiative. In order to ground discussion, we describe our current dialogue system, the Duke Programming Tutor. Through this system, we present a temperature-based model and algorithm which provide a basis for making decisions about dialogue focus and initiative.

## Tutorial Dialogue

Humans engaged in tutorial sessions follow different conversational conventions than they would in many other forms of dialogue. Tutorial dialogues are similar to the apprentice-expert or task-oriented dialogues in which the expert has complete knowledge of how to accomplish the task and guides the novice through the process (Grosz & Sidner 1986; Carberry 1990). In a tutorial domain, the tutor has all the knowledge about the subject area, and is only unsure of what the student knows.[1] If we are training someone to accomplish some task, we can view the problem as a traditional task-oriented domain, with the apprentice/student attempting to achieve a task (debug a program, fix a car) with assistance from the expert/tutor. While this "learning by doing" is one pedagogy, it is certainly not the only way the expert/tutor might want to teach.

In particular, a drawback to thinking about tutoring as task-oriented is that when the task is achieved, we don't necessarily know anything about how much the apprentice has learned — they may have just followed the expert's instructions without internalizing anything. In the extreme, we can imagine a misguided task-oriented system that when initially asked "What do I do?" simply offers a complete solution to the student. When presented with a similar question, human tutors use leading questions and vague answers in order to elicit responses from students. Figure 1 shows an example[2] of a human tutor using questions in helping a student learn to write a program. In the tutorial dialogue, the goal is not to simply *achieve* a correct program, but is rather the more abstract goal of ensuring that the student understands *how to achieve* the task. This type of interaction (in which the tutor is less responsive and helpful than possible) would still be viewed as proper discourse by the participants because it may better achieve a longer term goal.

| | |
|---|---|
| T: | So let's say we're going to write a program that says "I can write a computer program". OK? |
| S: | *OK.* |
| T: | So that's the text we want it to print to the screen. What's the first thing we need to put into the program? |
| S: | ***need to write the program name.*** |
| T: | Right. OK. What else do we need along with the name? |
| S: | ***program and then name.*** |
| T: | Right. OK. Let's go ahead and put that in. |

Figure 1: A sample human-human tutorial dialogue

A second challenge in tutorial discourse is the increased opportunities for misunderstanding. In the course of tutorial dialogue, we must assume that the student could be confused and may offer contradictory evidence of understanding. They might say they understand when they do not, or guess at a correct answer. Correct user modeling of understanding is vital to building a good tutor(Kass 1989). Of course, in a

[1] A tutor that can learn new information or new pedagogies from an expert is possible, but beyond the scope of this discussion.

[2] This excerpt is taken from a human/human tutorial session conducted in our lab.

spoken language dialogue system we will have to contend with additional uncertainties introduced by ambiguous utterances and erroneous speech recognition. In particular, we wish to be able to model both student understanding levels for various concepts and our confidence in these levels. These levels could be the result of both direct observations and derived evidence. Ideally, we would like to be able to model and reason with propositions such as "I'm pretty sure that the user does not completely understand what concept X is."

Another key difference between a tutorial domain and other task-oriented domains is the way that the system should "engage" the user. In many typical voice domains, the system need only respond directly to a user's utterance. Studies have determined that efficient teaching requires the teacher to be attentive and animated (Norton 1983). *Engagement* is the set of actions undertaken by the system to demonstrate its commitment to the success of the dialogue. In human/human tutoring dialogues, engagement includes actions such as head nods, "uh huh's", and quizzical looks when the student's work starts to go astray. We have come to call engagement those actions dealing with pro-activity and human-like non-verbal communications.

Both the need to model understanding and uncertainty and the necessity of engaging the student affect how initiative is determined in a tutorial dialogue. Since we are expecting some students to be true novices, with little or no prior knowledge of the domain, the tutor should be able to seize initiative, guiding the dialogue by providing needed tutorial assistance. However, if the student is more advanced, or as the student learns, the student should be able to direct the conversation by asking specific questions (in this case, the tutor more closely follows the standard apprentice-expert model). Thus, a successful tutorial dialogue system must support mixed-initiative, allowing both participants to control the discussion at different points. One can imagine a pedagogy in which more or less initiative is granted to the student, depending on the tutor's perception of the student's progress. If the student appears to be confused, the tutor may try to be more directive in leading the discussion.

## The Duke Programming Tutor

In order to explore tutorial dialogue, we have built a system to assist students in learning to write simple programs. The Duke Programming Tutor (DPT) is a multi-modal, voice dialogue system. In the early stages of introductory computer science classes, students are often unable to successfully write and debug their programs without the aid of a teaching assistant. Our goal is to lead a student through completing a standard programming lab from the course. There have been numerous programming tutor systems studied and built (e.g. (Johnson & Soloway 1984), (Anderson & Skwarecki 1986)). However, a primary focus of our project is the role of spoken dialogue in tutorial interaction. Currently, the system as implemented can support the input of voice, text (the student's program), and mouse selection, and can output combinations of voice, text and graphics. The system can provide both tutorial and debugging information to the student.

## Feature Vectors for Dialogue

In early versions of the DPT, based on the work of (Smith, Hipp, & Biermann 1995) and (Guinn 1995), we could not control initiative based on the student's understanding level or the needs of engagement. One method we are currently exploring attempts to address this shortcoming. Domain knowledge is represented in a large semantic network, where each node maps to a particular concept or piece of syntax. We attach a *feature vector* to each node that is used to determine which node will become the next focus, which will determine the next topics discussed.

There are a number of features that could be useful in determining the next focus. Certainly it depends on student understanding and our certainty in it, student interest, and pedagogical importance. We might want to subdivide understanding and certainty into *direct* (what a student tells us and what we observe in the program) and *derived* (evidence that propagates from other nodes). In addition, we could have a feature that encodes how many times this node has been discussed with the student. A node could also have a feature which is the distance between the it and the focus. This *focus distance* feature can be thought of in the sense of semantic differences after (Carberry 1990). Since we want to make progress towards having high confidence that the user understands the highest level construct ("student understands how to write a simple program"), we should also encode some form of goal derivatives — how changing a feature of a given node will affect a corresponding feature of the goal. If our system has initiative, our dialogue focus could then be driven by choosing the node that has the greatest affect on the goal node. Figure 2 is a table that summarizes the possible features we have discussed.

## Using Temperature to Determine Focus

Assuming that we have a semantic network with feature vectors at each node, we now discuss how we might use this to aid in directing the flow of a dialogue. In the vector model outlined above, we have the possibility of

**Derived Truth** The degree to which the node is true, as evidenced by connected nodes.

**Derived Certainty** The degree to which we are certain of the truth value, as evidenced by connected nodes.

**Direct Truth** The degree to which the node is true, as evidenced at this node.

**Direct Certainty** The degree to which we are certain of the truth value, as evidenced at this node.

**Truth Goal Derivative** The degree to which changing this node's truth level will positively affect the goal's truth level.

**Certainty Goal Derivative** The degree to which changing this node's certainty level will positively affect the goal's certainty level.

**Focus Distance** The minimum arc distance from this node to the current focus node.

**Goal Distance** The minimum arc distance from this node to the goal node.

**Repetition Cost** The number of times this node has been the focus.

**Interest** The user's interest in this node.

**Importance** The relative importance of this node.

Figure 2: Possible Features

a continuous spectrum of initiative. For example, the system's goal might be to raise truth and confidence levels of some node above some predetermined threshold. In order to accomplish this, we can choose the node with the greatest goal-derivative at each step as the focus of the dialogue. This choice gives the system full initiative — it pays no attention to the student's wishes. However, what if the student indicates interest in some concept? This may raise the interest feature of a set of nodes, perhaps "far" from the current focus. If the student is granted initiative, we could choose the node with the greatest user interest as the focus, without regard for how it would affect the root goal or how far from the current focus it is. Of course, all points in between these two extremes are interesting, in which there are tradeoffs between what each of the participants thinks is the best way to proceed.

We call the above approach to determining focus the

While the goal's condition is not satisfied, repeat:

1. Compute the temperature of each node.
2. Choose the focus to be the node with the maximum temperature.
3. Perform some communication about the focus with the student.
4. Update the focus node values (e.g. increase repetition count, decay student interest).
5. Update feature vectors based on the environment and student responses.
6. Use a propagation function to adjust feature vectors of adjacent nodes.

Figure 3: An algorithm using temperature-based focus.

*temperature-based approach to dialogue.* Let $f_{temp}(v_i)$ be a linear function that combines the $m$ real-valued features of node $i$. Note that there is nothing that would require the temperature function be a linear combination of weights. We could, for example, train a neural network that, given a vector, produces a temperature as output. We have a real-valued weight $w_j$ associated with each feature type, indicating the importance each should have in deciding what to do next. Under these assumptions, the temperature function might be $f_{temp}(v_i) = \sum_{j=1}^{m} w_j v_{ij}$. We can then set the next focus to be $max_{1 \leq i \leq n} f_{temp}(v_i)$ if there are $n$ nodes in the current network. With this temperature function, it should be clear that we can achieve some of the above initiative behaviors by changing the weights to reflect what we should pay more attention to — the user's interest or the effect on a root goal. Of course, many more subtle weightings are possible to achieve other interesting dialogue behaviors, including such things as preferring to not change topic too quickly or too often, avoiding repetitiveness, stressing differences in derived versus direct evidence, and doing pedagogically important things first. Figure 3 outlines an algorithm for using temperature to guide tutorial dialogue. The propagation and update functions mentioned are a fundamental part of this algorithm, and while we know some properties they must have, we are still experimenting with various forms these might take.

## Engagement in a Temperature-based Model

The implementation of a system which can actively and realistically engage the user provides a key element for effectively using mixed-initiative dialogue: a way to communicate the level of initiative to the user. Consider a tutorial dialogue between two individuals with the current focus centered on programming construct X. Suddenly, the user initiates an utterance about construct Y. Depending on the system's willingness to grant the user initiative, a choice must be made to maintain focus on X or switch to Y. A globally cognizant system will be aware of the switch and might reply with an explanation explaining why the focus decision was made. A system might speak "OK, we will do Y, but then return to X." A globally cognizant system *with engagement* will be able to express the same information without the need for a separate utterance. For example, it might display a facial expression in conjunction with head nods which conveys "I really don't want to talk about Y, but here is the answer."

Engagement also requires that the initiative is subject to change between utterances, which is in contrast to systems in which initiative decisions are made at each turn (Guinn 1996). As a example, suppose that the system completed a detailed explanation about a particular programming construct which the user has incorrectly used. The system would expect the user to fix her error or ask for additional help. After a certain amount of time has elapsed with neither event occurring, an engaging system might ask the user if she requires additional help. The amount of time the system waits before initiating additional utterances is a function of the history of the dialogue and the current user model as expressed in the feature vectors.

## Conclusion and Future Work

In order to explore whether feature vectors can aid in managing dialogue, we have incorporated the above ideas in a limited way into the Duke Programming Tutor. The system uses a simple linear temperature model in order to determine focus and initiative for the dialogue. Engagement is accomplished with a digitized face capable of indicating various facial expressions. We tested this system on 13 students from the introductory Computer Science class at Duke in September. We are still examining the results of this experiment. Though there are many unresolved issues, we think that in a tutorial domain, feature vectors may be an important tool in building an effective mixed-initiative tutorial dialogue system.


## Acknowledgements

This research is supported by the Office of Naval Research grant N00014-94-1-0938, the National Science Foundation grant IRI-92-21842 and a grant from the Research Triangle Institute, which is funded in part by the Army Research Office. Other individuals who have contributed to the Duke Programming Tutor include Curry Guinn, Zheng Liang, Phil Long, Douglas Melamed and Krishnan Rajagopalan.


## References


Anderson, J., and Skwarecki, E. 1986. The automated tutoring of introductory computer programming. *Communications of the ACM* 29:842–849.

Carberry, S. 1990. *Plan recognition in natural language dialogue.* ACL-MIT Press series in natural language processing. Cambridge, Massachusetts: MIT Press.

Grosz, B. J., and Sidner, C. L. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics* 12(3):175–204.

Guinn, C. I. 1995. *Meta-Dialogue Behaviors: Improving the Efficiency of Human-Machine Dialogue–A Computational Model of Variable Inititive and Negotiation in Collaborative Problem-Solving.* Ph.D. Dissertation, Duke University.

Guinn, C. I. 1996. Mechanisms for mixed-initiative human-computer collaborative discource. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, 278–285.

Johnson, W. L., and Soloway, E. 1984. Intention-based diagnosis of programming errors. AAAI.

Kass, R. 1989. Student modeling in intelligent tutoring systems–implications for user modeling. In Kobsa, A., and Wahlster, W., eds., *User Models in Dialog Systems.* Berlin: Springer-Verlag. chapter 14, 386–410.

Norton, R. 1983. *Communicator Style.* Beverly Hills, CA: Sage Publications.

Smith, R. W.; Hipp, D. R.; and Biermann, A. W. 1995. An arctitecture for voice dialog systems based on prolog-style theorem proving. *Computational Linguistics* 21(3):281–320.