BROADENING INPUT UNDERSTANDING IN A LANGUAGE-BASED INTELLIGENT TUTORING SYSTEM

BY

MICHAEL S. GLASS

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science
in the Graduate College of the
Illinois Institute of Technology

Approved_____
Adviser

Chicago, Illinois
May, 1999

ACKNOWLEDGMENT

I would like to express my gratitude to my advisor, Dr. Martha W. Evens, who enrolled me in the Ph.D. program before I knew that I wanted to be a student again, then suffered my presence with patience and wisdom for too many years.

I thank Dr. Joel Michael and Dr. Allen Rovick of Rush Medical College, who together with Dr. Evens created and ran the CIRCSIM-Tutor project in which I have labored so happily. They give of themselves unstintingly.

I thank my professors, fellow students, and researchers on this project, who provided me which so much of my intellectual life since I came here and tolerated my disordered papers and books all over the lab. Let me particularly mention those with whom I spent so many hours in conversation: Peter Greene, Ramzan Ali Khuwaja, Kumar Ramachandran, Murugan Kannan, Mohammad Elmi, Greg Hume, Greg Sanders, Farhana Shah, Stefan Brandle, Yujian Zhou, and Jung Hee Kim. It has been truly a delight to work here.

Finally I particularly thank my mate and co-worker, who stayed up late so many nights, Reva Freedman.

This work was supported by the Cognitive Science Program, Office of Naval Research under Grant No. N00014-94-1-0388, to the Illinois Institute of Technology, and the associated AASERT Grants to the Illinois Institute of Technology. The content does not reflect the position or policy of the government and no official endorsement should be inferred.

M. S. G.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENT iii

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

I. INTRODUCTION 1

Outline of This Thesis 3

II. DESCRIPTION, HISTORY, AND FUTURE OF CIRCSIM-TUTOR . 5

Description of CIRCSIM-Tutor 5
History of the CIRCSIM-Tutor Project 12
CIRCSIM-Tutor Version 3 13
History of CIRCSIM-Tutor Input Understanding 15

III. THE IMPORTANCE OF STUDENT LANGUAGE IN TUTORING 20

Education Studies on the Importance of Language and Tutoring . . . 21
A Difference in Style Between Novice and Expert Tutors 24
Recent Developments in Similar Intelligent Tutoring Systems 29

IV. OTHER DIALOGUE-BASED INTELLIGENT TUTORING SYSTEMS 33

SCHOLAR: Early Intelligent Tutor with Mixed Dialogue 33
Creanimate: Combining Socratic Tutoring with Case-Based Techniques 33
Dialogue-Enhanced Explanation Systems 35

CHAPTER Page

V. PHENOMENA ENCOUNTERED BY THE INPUT UNDERSTANDER 38

Qualitative Changes 39
Phrasal Answers 43
Spelling Errors 46
Hedges 48
Category Errors 50
Near-Misses and Other Appropriate but Unexpected Answers . . . 51
Student Initiatives 54
Equations 56
Persuading Students to Use Our New Features 57

VI. TECHNOLOGICAL BACKGROUND TO THE INPUT UNDERSTANDER 60

Semantic Grammar in SOPHIE, an Early Free-Text Input ITS 60
Message Understanding 64
More Recent Approaches to Symbolic Understanding 68
Direct Memory Access Parsing 72

VII. USING LATENT SEMANTIC ANALYSIS IN THE INPUT UNDERSTANDING TASK 75

Simplified Introduction to LSA 75
LSA in More Detail 77
Latent Semantics 79
LSA and Psychology 82
Application of LSA in AutoTutor 87
Diagnostic Questions: A Potential Application of LSA in CIRCSIM-Tutor 91
Experiment using LSA for CIRCSIM-Tutor 95

VIII. CIRCSIM-TUTOR V2.5 ENHANCED INPUT UNDERSTANDER 99

Context of the Input Understander in CIRCSIM-Tutor v. 2 99
Overview of Processing 102
Finite State Transducer Mechanism 105
Some Finite State Transducers 107
Lexicon and Lexical Lookup 111
Spelling Correction 114

CHAPTER Page
Ontologies 118
Performance with a Class of Medical Students 118

IX. MORE LANGUAGE-ORIENTED TOPICS FOR INPUT UNDERSTANDING 122

Spelling Correction and the Input Understander 122
Use of Equations in the Transcripts 124

X. CONCLUSIONS 135

Summary and Significance 135
Future Work 137

BIBLIOGRAPHY 138

LIST OF TABLES

Table		Page
1.	Who Ultimately Corrects the Error	29
2.	Short Answers to "What is the Value of ...."	40
3.	Responses to "Where are you going?"	41
4.	Some Attested Spelling Errors	46
5.	Sentences Closest to "Frank Starling"	96
6.	Values of the MEAN Attribute in the Lexicon	114
7.	Input Understander Performance in November 1998	119
8.	Representative Samples of Input Understander Failure in November 1998	121
9.	Representative Spelling Errors Corrected in November 1998	121
10.	Identifying Equations	125

LIST OF FIGURES

Figure		Page
1.	Top Level Concept Map	7
2.	Schematic View of the CIRCSIM-Tutor Screen	10
3.	Extract from Marked-up Transcript	16
4.	CST v. 2 Questions, Logic Form(s) and Input Understander Results	101
5.	CST v. 2 Improved Input Understander Processing	103
6.	Example Finite State Transducer: Copula Deletion	108
7.	Mechanism Answer Ontology	117

CHAPTER I

INTRODUCTION

This work is about improving the input understanding component and the general quality of the tutorial dialogue for CIRCSIM-Tutor. CIRCSIM-Tutor is an intelligent tutoring system intended to teach medical students about a negative feedback loop that regulates blood pressure in the human body. It has been amply described in numerous other IIT documents so I will keep the description here rather short.

For the purpose of understanding this work, it is important to know that CIRCSIM-Tutor is language-based. It instructs by carrying out a dialogue with the student in English. There are no diagrams, hypertext links, multiple-choice questions, or pull-down menus. Except for a chart the students must fill in to record their predictions, the students spend their time typing black letters on a white screen and the tutor replies in kind. To a large extent the designers of this intelligent tutoring system were motivated by the belief that the use of language enhances learning.

My work is oriented around the language issues in CIRCSIM-Tutor. I have replaced the input understanding component of the existing version 2 of CIRCSIM-Tutor, which is now in use at Rush Medical College. This new input understander, with some enhancements, will be incorporated in the new version 3 which is under construction. I have also been involved with language issues throughout the project, working on surface-level generation, the language of tutoring tactics, the lexicon, verb case frames, the internal representation of various linguistic objects, transcript markup and analysis, and so on.

Our primary source of data on both student and tutor behavior is the set of keyboard-to-keyboard tutoring transcripts that the CIRCSIM-Tutor project has accumulated. We have more than eighty of them, most one or two hours long, including fifty with expert tutors. In this thesis I sometimes use raw transcript extracts, and sometimes cleaned-up extracts with fixed spellings and expanded abbreviations, in order to illustrate the behaviors we want the input understander to handle. An example reference to a transcript extract would be "K51-tu-17-3", meaning transcript K51, tutor utterance, turn number 17, sentence 3. I often abbreviate this to simply "K51-17" meaning an excerpt starting with turn 17 of transcript K51. In the fifty transcripts numbered K1 through K51 (K29 is missing) the tutors are Joel Michael and Allen Rovick, principal investigators on the CIRCSIM-Tutor project and professors of physiology at Rush Medical College. In the thirty or so transcripts numbered N1 through N31 the tutors are "novice" tutors, medical students who were recruited and trained to varying degrees to teach their peers. In the text below I refer to the transcripts of expert tutors as the K-series and transcripts of novice tutors as the N-series.

I have also obtained some examples from logs of medical students using various editions of CIRCSIM-Tutor version 2. We have not indexed and numbered these logs, so these examples contain no references.

Version 1 of CIRCSIM-Tutor was completed in 1989 and version 2 was completed in 1992. In April and November 1998 physiology classes of first-year medical students at Rush used CIRCSIM-Tutor for the first time. The program that they used is an enhanced version 2 containing, among other improvements, the replacement input understander

described here. We informally call this version 2.5. Version 3 will be a total rewrite. Freedman [1996] has laid the groundwork for version 3, coming up with a new model for tutorial planning and dialogue generation.

The new input understander in version 2.5 is constructed to be robust, based on a philosophy of extracting from the student utterance only what is necessary for answering the tutor's question. Its processing model is finite state transducers which perform only minimal parsing.

For version 3 we would like to enhance the input understander primarily in two ways: an increased ability to recognize and handle various "near miss" student answers, and the ability to handle some student initiatives and the answers to some open-class questions. For the latter I plan to use the relatively new technology of Latent Semantic Analysis.

Outline of This Thesis

Here is an outline of topics to come:

- Chapter 2 will introduce the CIRCSIM-Tutor enterprise, its history and motivation, and the physiology involved. It includes a section summarizing the basic idea going into the building of the new version: an attempt to mimic human tutors' observed patterns of language and tactics.
- Chapter 3 contains a justification for doing language-based tutoring. It assembles some of the evidence that shows that tutoring is worthwhile and that forcing students to verbalize is worthwhile. CIRCSIM-Tutor is among the small number of tutor-driven intelligent tutoring systems which require the student's free-text input. The topic of

this thesis is how to cope with that textual input. It seemed to me that justification for this somewhat quixotic enterprise is required.

- Chapter 4 is a brief review of some other tutor-driven dialogue-based tutoring systems.
- Chapter 5 is a description of the phenomena which occur in student utterances in tutorial dialogue. Even the highly simplified conversational domain of CIRCSIM-Tutor presents some unexpected challenges.
- Chapter 6 is a review of some of the technologies, old and new, which might be applicable to the input-understanding task.
- Chapter 7 is devoted to the recent technology of Latent Semantic Analysis. I review the technique, show how it is being used in another intelligent tutoring system, and make a suggestion as to how we could use it in CIRCSIM-Tutor to handle some kinds of tutorial exchanges which are currently beyond CIRCSIM-Tutor's ability.
- Chapter 8 is a description of my new input understander for CIRCSIM-Tutor, along with the results of using it with physiology classes at Rush Medical College.
- Chapter 9 is a review of some further language phenomena and improvements which would make for interesting enhancements to the input understander.

Finally, of course, there are conclusions.

CHAPTER II

DESCRIPTION, HISTORY, AND FUTURE OF CIRCSIM-TUTOR

Description of CIRCSIM-Tutor

CIRCSIM-Tutor is a dialogue-based intelligent tutoring system designed to tutor the basics of the baroreceptor reflex, an important mechanism for blood pressure regulation. The intended audience is first year medical students in their physiology course. Its salient characteristics are:

- The fundamental instructional mode is the student working problems presented by the tutor.
- Most of the communication between program and student is via written text.
- The tutor almost always has control over the dialogue, telling the student to perform small tasks and asking questions of the student.

CIRCSIM-Tutor's treatment of the material is like this:

- The program uses a simplified model of the baroreceptor reflex, heart, and circulatory system designed to emphasize the negative feedback aspect of the regulation process.
- The negative feedback regulation process is modeled as three disjoint chronological stages called direct response, reflex response, and steady state.
- The student's basic task is to predict the qualitative changes (increase, decrease, no change) in the values of seven "core" physiological parameters.
- The student's problem-solving activity involves a series of steps: 1) read the problem statement, 2) predict the values of all seven variables for the first stage, 3) engage in

tutoring dialogue with the tutor until all errors are corrected, 4) repeat the prediction/tutoring cycle for the remaining two stages.

The student is assumed to have already been taught the material, and thus be familiar with the vocabulary and concepts as well as the simplified model and the structure of the problems. Prior mastery of the material is not assumed, but somebody who has never seen the material cannot reasonably expect to either learn much or perform well.

A lightning tour of the physiology and problem-solving will aid in understanding the tutoring and language examples to come. Figure 1, the "top level concept map," shows the important physiological notions including the seven core variables. The arrows show which variables affect which other variables; a plus or minus label indicates that the effect is directly or inversely proportional, respectively.

Start navigating at the top of the concept map with Central Venous Pressure (CVP), representing blood pressure in the great veins (or more particularly, that part of the veins known as the central venous compartment). The central venous compartment is like an expandable balloon, containing a reservoir of blood. At the beginning of the heartbeat cycle, blood flows from the great veins into the right atrium of the heart. Blood is not sucked in; rather, it is pushed in at low pressure while the heart is relatively quiet, causing the atrium to stretch out. The higher the value of CVP, the more blood is pushed in. Eventually the heart squeezes down, starting a beat and ejecting much of the blood which had flowed in. The amount of blood squeezed out in one beat is Stroke Volume (SV), so SV is directly proportional to CVP. SV is also affected by Inotropic State (IS),

a measure of neural stimulation of the heart muscle; increasing IS increases the amount of squeezing, thereby increasing SV.

With each beat, blood is ejected from the left ventricle of the heart. The model has a big simplification here in that the right atrium isn't actually connected to the left ventricle. Thus the model ignores several heart chambers plus the entire pulmonary circulation, but these are not needed for illustrating the baroreceptor reflex. Stroke volume

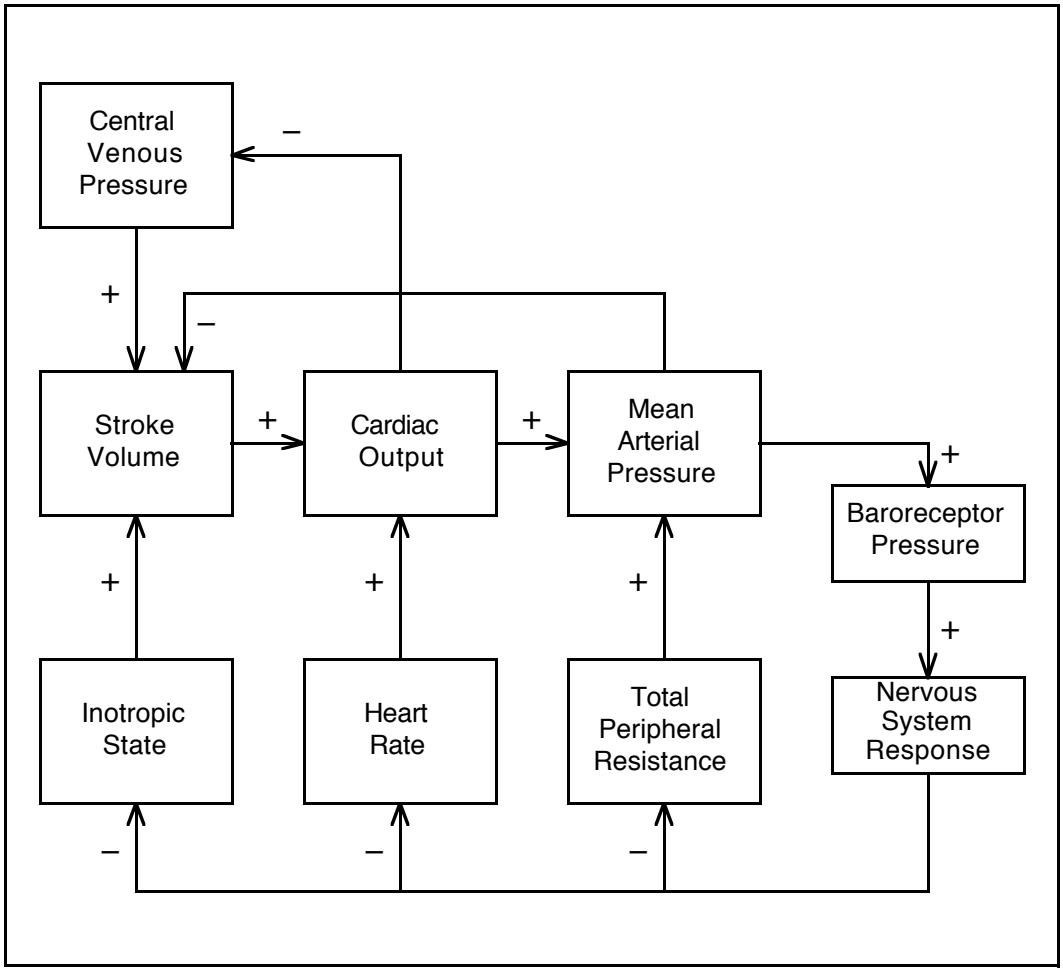

Figure 1. Top Level Concept Map

(liters per beat) times Heart Rate (HR) (beats per minute) gives Cardiac Output (CO) from the heart (liters per minute). HR is another neurally controlled variable.

The blood is being forced out at high pressure into the arteries. Here is the highest pressure in the system, as the blood is pushing against resistance downstream in the arterioles and then in the capillaries and veins. Mean Arterial Pressure (MAP) is the root mean squared arterial pressure (which oscillates up and down with each heartbeat), and Total Peripheral Resistance (TPR) is the sum of all the downstream resistance. Increasing CO (the amount being pumped) and TPR (the resistance the heart is pumping into) both increase MAP. TPR is the third parameter adjusted by neural stimulation.

In the first chronological stage, known as the Direct Response (DR), we assume that neural stimulation is held constant. Thus when presented with an abnormal situation affecting blood pressure, the three neural variables do not change but the other four do. To predict qualitative changes, just trace the arrows. For example if a person hemorrhages blood, this is assumed to decrease the amount of blood in the central venous compartment (the reservoir), decreasing CVP, which decreases SV (since IS stays constant), decreasing CO (since HR remains constant), decreasing MAP (since TPR remains constant), and we are done[1].

The initial change to the body is called a "procedure." Hemorrhage is, in fact, one of CIRCSIM-Tutor's procedures.

---

[1] The rules for this exercise are slightly complicated but can be summarized as 1) when given a choice, follow the arrows toward MAP first, then pick up any variable you missed, and 2) if the variable you are trying to predict has several determinants, and you know the values of some but not all, just ignore the ones you do not know. Freedman

The next stage is Reflex Response (RR), where the Baroreceptors, which are pressure sensors monitoring MAP, try to adjust the neurally controlled variables so as to bring MAP back toward normal. So in the case of the hemorrhage, where MAP is decreased in DR, the reflex response is to increase IS, HR, and TPR, all of which will tend to increase MAP. Then the other variables are predicted, using the same method as before. Predictions for RR are relative to the end of DR, so that if CO (for example) was down in DR then up in RR, we say "up," because during RR it increased.

The final stage is Steady State (SS), where you predict the qualitative differences from before the problem started until the body has stabilized. These differences are the sum of the changes in DR (the direct effect of the original problem) and RR (what happened by way of compensation). MAP never returns all the way to its original level. For accuracy, you should know that DR and RR do not necessarily represent observable chronological stages: it is not the case that first all the blood is lost in a hemorrhage, then later the reflex compensates. In reality it is all happening simultaneously. The stages are a way of separating the behavior into components.

Two important concepts for thinking about these problems are the "procedure variable" and the "primary variable." The procedure variable is the first physiological variable that students know about which is affected in DR. For the hemorrhage procedure it will be central blood volume or a related variable. The primary variable is the first variable in the prediction table (a core variable) which is affected in DR. In the hemorrhage case it is CVP. A policy of Joel Michael and Allen Rovick, which can be seen both in

---

[1996] formalized the whole set of rules.

human tutoring as well as CST v. 2 tutoring, is to make sure the student understands the correct primary variable before predicting any others.

This ends the quick introduction to the physiology and problem-solving in CIRCSIM-Tutor.

A schematic view of the CIRCSIM-Tutor screen is shown in Figure 2. On top is the procedure description, which is always at least partly visible (sometimes the bottom of the description scrolls off the window, but it can be retrieved). The other two windows are the prediction table, where the student enters qualitative predictions a stage at a time, and the dialogue window. The predictions are always visible; the student corrects them as they are tutored. Tutorial dialogue for the current procedure can be scrolled back for reference.

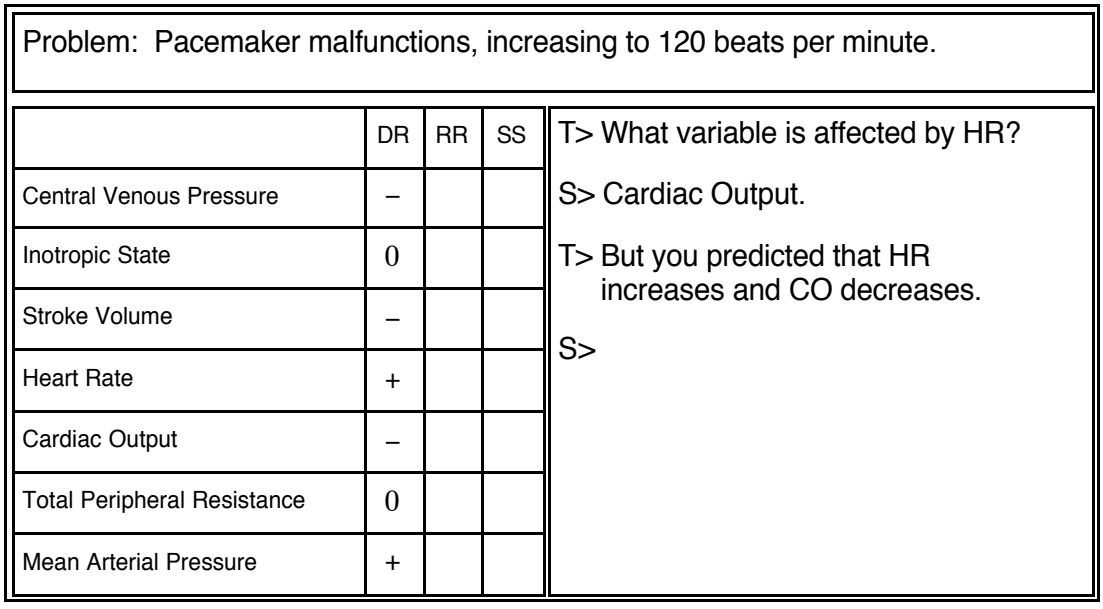

Figure 2. Schematic View of the CIRCSIM-Tutor Screen

The major components of CIRCSIM-Tutor v. 2 [Woo 1991] are a domain knowledge base with two associated domain problem solvers, a student model, a tutorial planner, a discourse planner, an input understander, and a text generator. One domain problem solver produces the answers for the full procedure; the other produces incremental answers to questions which arise during tutoring (for example, what is the dominant determinant in the case of a particular variable under discussion). The student model is responsible for judging the student's initial predictions as well as the answers to the tutor's questions. The result of the student model is not simply a list of wrong predictions. The student model is capable of noticing that important relationships between parameters were violated which might require tutoring. Recently, we have added the capability of distinguishing a number of instances of "near miss" answers which are not strictly right or wrong.

The tutorial planner in CST v. 2 runs after each stage is predicted, producing a plan with a few steps for correcting each error. The discourse planner attempts to execute the plan one step at a time. Each step is realized as possibly several actions, for instance a statement then a question. After a question the discourse planner interprets the student's answer before proceeding to the next step. The discourse planner calls upon the text generator, the input understander, and the student modeler. Since in CST v. 2 the tutorial planner runs once after all the predictions are made but the discourse planner runs after every student input, it is the discourse planner which handles the problem of changing the tutoring to fit the student's responses.

History of the CIRCSIM-Tutor Project

The immediate predecessor to CIRCSIM-Tutor is CIRCSIM [Rovick and Michael 1986, 1992], a program which introduced the same problems and physiological model as adopted by CIRCSIM-Tutor. It prompts the students for all the qualitative predictions at once (not one stage at a time), and has approximately 230 stored paragraphs which it uses for tutoring the mistakes. CIRCSIM is able to detect multi-variable patterns of errors, and thus able to address common underlying student misconceptions. One feature is that the first procedure is guided by the tutor, solved in detail a step at a time. CIRCSIM's capabilities for dialogue are much more limited than CIRCSIM-Tutor's. CIRCSIM is still in use by the physiology classes at Rush Medical College as a regular part of the curriculum.

Before CIRCSIM there were several teaching aids that incorporated a quantitative mathematical model, first MACMAN [Dickinson et al. 1973], then HEARTSIM [Rovick and Brenner, 1983]. MACMAN is a quantitative model. Students were expected to make predictions, then use the model to verify their results. However students often didn't know enough to successfully design their own experiments, and they depended on skilled instructors to interpret the results. HEARTSIM is the same model with some didactic software added. It has a predefined set of procedures for the students to run. Student predictions are made and scored qualitatively, and the software walks the students through a sequence of pedagogical steps such as finding and correcting logical errors in the predictions before running the model.

What prompted the CIRCSIM-Tutor project was the observation that too often students who successfully completed CIRCSIM still retained serious misconceptions.

Michael and Rovick felt that having a conversation with the student might provide the opportunity to detect and remedy these misconceptions. Additionally, they were convinced that more active student involvement will lead to better learning.

CIRCSIM-Tutor Version 3

There have been several proposals [Khuwaja 1994, Freedman 1996] for the planning architecture of the next version of CIRCSIM-Tutor, version 3. Freedman's is the one we are using.

It is Freedman's observation that the planner which runs an intelligent tutoring system can be thought of as primarily generating a dialogue. Thus CIRCSIM-Tutor version 3 is not, at its heart, a model-tracing tutor [Anderson 1993, pp. 235–255] which causes the student to follow a trace of the problem solution. It is true that the overall dialogue follows, in outline, a trace of parts of the solution procedure (those parts of it that the student got wrong in the initial predictions), but on a more local level we are trying to generate tutorial dialogue which is informed less by the needs of the problem trace and more by the actual dialogue patterns that people use for tutoring.

The actions of the new CIRCSIM-Tutor will be based on tutoring patterns we have observed in the transcripts. It has a catalog of conversational gambits (tutoring tactics and methods) coded as planning schemas. In order to satisfy a planning goal (such as, ensure the student is aware of the right answer to some issue) it tries the available gambits until the student demonstrates the correct knowledge.

The observation which underlies this new model is that to a large extent tutoring methods are a language phenomenon. If a student doesn't understand some point, a good

tutor often has several different ways to ask or explain the same issue. If a method involves a multi-part explanation, the tutor will pick and choose which parts to say outright and which to elicit from the student conversationally. We cannot imagine deriving from detailed principles the bag of conversational tactics which a good tutor has available. Here is an example of a tutoring method we loosely call "move forward." There are several versions of move forward. In this one the tutor stipulates some variables which have changed (possibly as the result of the preceding discussion), then asks what changes directly follow. In this excerpt that tutoring method is applied twice in a row:

T: If HR goes down, what effect would that have?
S: It would cause a decrease in CO.
T: Yes. And if CO goes down and TPR goes down, what effect do they have?
S: They cause decreased MAP. [K43-120]

Contrast that with "tutoring by determinants," in which the pattern is to first make sure the student knows the determinants of a target variable which needs to be tutored, then make sure the student knows how those determinants have changed, then find out the change in the target variable. Here is an example:

T: Now, what two parameters in the predictions table together determine the value of the SV?
* S: CO and HR
* T: No.... What I was asking is what determines how much blood is ejected from the heart each time it beats (the SV)?
S: RAP and CC
T: Good. Well, you made predictions about how RAP and CC would change as a result of the pacemaker malfunction. What do you think will happen to SV?
S: ... [K14-47]

The starred turns in this example illustrate another reason why we like to think of tutoring as a linguistic process. The student gave an incorrect answer to the first question. The tutor's response was to re-ask the *same* question, rephrased. In this case the tutor simply inserted a definition of stroke volume, viz.: "how much blood is ejected from the

heart each time it beats." It seems clear that the rephrasing of a question is a linguistic process. Simply knowing that rephrasing the question is a productive gambit in this case is a kind of pedagogical or linguistic knowledge, certainly not physiological knowledge.

This example of rephrasing also illustrates a linguistic tutoring tactic I call "evocative language." I claim that the student almost certainly knew the definition of stroke volume. My hypothesis is that in rephrasing the question the tutor chose words which evoked useful images in the student's mind. We have similar examples in the transcripts where the tutor used the word "afterload" as a synonym for "mean arterial pressure" to represent the pressure the left ventricle is pumping against.

Deriving tutoring behavior from the transcripts is quite a serious effort. A central activity is to analyze the goal structure of the tutoring sessions in great detail and mark up the transcripts. Using Freedman's initial plan, Kim, Freedman, and occasionally others (including myself) have been marking up substantial parts of the K-series transcripts, using the SGML markup language. Figure 3 shows an extract from a marked-up transcript from Kim. Kim et al. [1998] show how the contents of marked-up transcripts are being used to describe the sentences which will be produced by the CIRCSIM-Tutor v. 3 text generator. Freedman et al. [1998a, b] show how machine learning can be applied to various aspects of the marked-up transcript to derive rules for tutoring behavior.

History of CIRCSIM-Tutor Input Understanding

Two of my predecessors have written input understanders for earlier versions of CIRCSIM-Tutor, Yoon-Hee Lee [1990] and Jai Hyun Seu [Seu 1992, Seu and Evens 1991]. Lee's code was written originally in Lisp to run in isolation on a Xerox artificial

```
<T-corrects-variable  var=SV>
 <T-tutors-variable>
  <T-moves-forward>
   <T-tutors-consequence-value>
    <T-informs  from-var=CO  from-value=decrease  to-var=SV>
        K44-tu-152-2: So, you predicted CO D
    </T-informs>
    <T-informs  from-var=CVP  from-value=increase  to-var=SV
    DM="and,now">
        and now say that RAP must I.
    </T-informs>
    <T-elicits>
        K44-tu-152-3: What then happens to SV?
     <S-ans  catg=correct>
        K44-st-153-1: Increase because of inc. filling.
     </S-ans>
     <T-ack  type=positive>
        K44-tu-154-1: Exactly.
     </T-ack>
    </T-elicits>
   </T-tutors-consequence-value>
  </T-moves-forward>

  <T-explores-anomaly>
   <T-presents-anomaly>
    <T-informs  info=det-value  DM="so"  narrative-
    mode=reference>
        K44-tu-154-2: So, you have now predicted that CO will fall
                      and that HR is down but SV is up.
    </T-informs>
   </T-presents-anomaly>

   <T-tutors-anomaly>
    <T-elicits>
        K44-tu-154-3: How is this possible?
     <S-ans  catg=correct>
        K44-st-155-1: If the HR decreases more than the SV
                      increases.
     </S-ans>
     <T-ack  type=positive>
        K44-tu-158-1: Exactly!
     </T-ack>
  ...etc...
</T-corrects-variable>
```

Figure 3. Extract from Marked-up Transcript

intelligence workstation. It was later integrated into CIRCSIM-Tutor on the Macintosh by Seu. This is the code that I replaced.

One of Lee's lasting contributions is his spelling correction algorithm. It lives on (as significantly updated by Mohammed Elmi [Elmi 1994, Elmi and Evens 1998]) in the new CIRCSIM-Tutor input understander. Chapter 8 contains a discussion of the spelling correction algorithm.

Lee and Seu's input understander was based on a Lexical-Functional Grammar parser (for a definition of LFG see Bresnan and Kaplan [1982]). This parser started with the student's utterance and produced a functional representation (called an "f-structure" in LFG parlance). The functional representation was converted by a piece of *ad hoc*-ery to the logical forms needed by the tutorial parts of CIRCSIM-Tutor.

Early in my work on this project, I also thought that LFG was the way to approach student input. I produced an LFG grammar of short student answers. The human-readable version of Lee's LFG grammar is lost, but comparing the bits in his thesis and code to my own LFG grammar reveals a significant difference, to wit: Lee regards a student's short answer as, syntactically, a kind of reduced sentence. I do not. However in a system such as Lee's this distinction would have made little practical difference[2]. The descriptions of reduced sentences he was producing were, perforce, fairly short; there is only so much you can say about a one or two word utterance.

---

[2] One practical difference showed up when I was modifiying Lee's parser to handle the parameter Inotropic State, abbreviated "IS." Since fragments were taken as reduced sentences, the program mechanically inserted the copula "is" in the course of parsing, causing conflicts with the new parameter name.

Seu's input understander is built upon Lee's. Seu took the spelling corrector basically intact, and upgraded both the parsing procedure and the procedure for producing the final logical form. The grammar was still LFG, producing f-structures. One of Seu's contribution's was basic handling of ellipsis and the pronoun "it." If the question asked of the student was "what is the value of cardiac output?" the student could answer:

a) CO increases
b) It increases
c) Increases

Answer a) is a complete sentence, b) contains a pronoun reference to "CO," and c) is an elliptical utterance where "CO" is assumed. Seu updated the input understander to handle some limited forms of student initiative, namely "I don't know" and requests for a definition, such as "What is CO?" and "I am confused about CO." I have included the "I don't know" facility in the new input understander, but not the requests for definition.

Let me note parenthetically that Seu also counted various characteristics of the transcripts which had been collected up until that time [Seu et al. 1991]. He has good statistics about things like sentence length and certain types of word usage. He also attempted to count questions, but since questions are not always marked (e.g. with a question mark or an obvious inversion of an auxiliary verb), and since a question mark sometimes signals a hedge and not a question, one would need to evaluate his procedure before using his answers. For the most part this analysis does not affect the design of the new input understander, but it could well be valuable for other purposes.

There are a number of factors which militated against preserving the input understander code from CST v. 2:

- I implemented other, more robust technology.
- The basic CST paradigm and data structures are being replaced for CST v. 3, making it hard to match the old input understander to the new task.
- New types of input, such as near misses, would have been difficult to recognize in the existing structure.
- The old input understander was a mess to read and maintain, having been written, translated and modified by a number of different people, some of whom were uncomfortable with English.

CHAPTER III

THE IMPORTANCE OF STUDENT LANGUAGE IN TUTORING

CIRCSIM-Tutor is lonely. Very few tutor-directed intelligent tutoring systems have been able to accept free-text input from students. I make a distinction here between systems where the student has the initiative and systems where the tutor has the initiative. CIRCSIM-Tutor is not a question-answering system where the student asks questions and the software provides answers, rather it is actively controlling the conversation with a tutorial agenda. At the 1996 Intelligent Tutoring Systems conference, the premier conference in the field, there were no papers on the problems of free-text student input in tutor-driven Intelligent Tutoring Systems [Frasson et al. 1996].

Given this state of affairs, it seemed to me that the CIRCSIM-Tutor enterprise needs some justification. The goal of this chapter is to explore the reasons for having the students type free-text answers, as opposed (for instance) to having them pick an answer from a menu. This issue cannot be comprehensively addressed in a single chapter, but I assemble evidence from a variety of sources.

- There are studies that show that vocalizing and giving explanations aids learning.
- There is evidence in the education literature that human tutoring is highly effective compared to other forms of instruction.
- As evidence that talking is beneficial, we have data that expert tutors are more likely than novice tutors to force the student to enunciate an answer, as opposed to having the tutor enunciate it.

- For various reasons which might also be relevant to CIRCSIM-Tutor, a number of other important intelligent tutoring system projects have recently chosen to handle free-text input.

Though it does not constitute a definitive justification for writing a free-text input Socratic tutoring program, this assemblage makes the CIRCSIM-Tutor project seem more worthwhile.

Education Studies on the Importance of Language and Tutoring

There is reason to believe that merely making students talk (or write) has value. Independent of whether you understand what they say, making them say it can be beneficial.

Chi and her collaborators have been conducting a series of experiments on what they call the "self-explanation effect." In one experiment, Chi et al. [1994] showed that prompting students to explain their understanding of the material they just studied is pedagogically useful. In that experiment eighth-grade students read aloud a 101-sentence passage about blood circulation. After every sentence the student was prompted to explain it to an experimenter. Most of the prompting questions were deliberately vague, asking the student to simply explain what was just read, although there were more specific prompts salted among the general ones. The control group read the text several times without explaining it out loud. (They had to read it several times so as to spend approximately the same amount of time with the text as the prompted students spent.) The explain-out-loud group showed considerably more improvement in understanding, as measured by pre- and post-tests. The experimenters went further in their analysis,

ranking the students on a scale of “high explainer” to “low explainer” and categorizing their utterances. Then they correlated the self-explanation behavior of individual students with the misconceptions evinced by the same students on the post-test. The results were unambiguous: prompting people to explain themselves increases their learning considerably.

A more recent experiment by Coleman et al. [1997] compared the learning performance of 84 undergraduates reading texts on several theories of evolution. Here the students were divided into three groups: students who merely heard the text read to them, students who read the text for themselves but were then asked to explain (or summarize) it, and students who read the text knowing that they would have to explain (or summarize) it to others. Again, explanation proved to be a powerful tool for learning.

Graesser [1993] conducted a study which dissected in detail 66 one-hour human tutoring sessions, in two different sets. One set consisted of 44 sessions of undergraduates being tutored for their psychology research methods course, using tutors who were students who had done well in the graduate course. For this group there were pre-tests and post-tests to determine how well students were learning. Another set consisted of 22 sessions of seventh-graders being tutored in seventh-grade algebra, using tutors who were high-school students with experience tutoring the subject. Among the primary results are that tutoring is, in fact, very effective. Furthermore this happens despite the fact that his tutors rarely used sophisticated strategies, such as Socratic dialogue. Furthermore the tutors spent little time diagnosing and addressing student misconceptions. (They did ask a good many diagnostic questions; I will address that issue

later in Chapter 7.) Graesser also showed that often his tutors were simply following scripts.

So what accounts for the success of Graesser's tutors? He discovered that even though the tutors had control over the conversation the students were asking approximately eight "deep reasoning" questions per hour of tutoring, where deep reasoning questions were defined as one of the following (excerpted from [Graesser 1993, p. 8]):

- Antecedent questions. What caused or justified an event or proposition?
- Consequence questions. What are the consequences of an event or proposition?
- Goal orientation. "Why" questions.
- Enablement. What state or event allows some other state or event to occur?
- Instrumental/procedural. What allows some agent to accomplish some goal?
- Expectational. "Why not" questions?

The tutors were asking these deep reasoning questions at a higher rate than the students. Altogether, tutoring sessions generated deep reasoning questions at a much higher rate than in normal classroom teaching. It was Graesser's opinion that these questions, not specific tutoring tactics or diagnostics, that accounted for much of the learning.

Cohen et al. [1982] conducted a metastudy of 65 controlled evaluations of school tutoring programs, concluding that tutored students outperform control subjects to a high degree of statistical significance. They culled the studies from a pool of 500 reported controlled studies in the literature, selecting only those that were conducted in actual

elementary and secondary school classes, were free from methodological flaws which might invalidate comparisons between the test group and the control group, and included quantitative measurement of outcomes. Sometimes the outcomes measured did not include academic achievement, leaving 52 studies with measures of achievement. In these 52 studies the median tutored student performed at the 66th percentile of the untutored students. Another way of stating the result is that tutoring raised performance by about two-fifths of one standard deviation, in whatever units were being measured. While these gains may not be large, they come from such a large data set that they are probably robust.

Bloom [1984] reported startlingly good results comparing classes of elementary school students taught in a normal classroom, in a normal classroom using "mastery learning," and tutored individually or in groups of up to three together. There were approximately 30 students in each category, and the experiment was limited to certain subjects (as opposed to the whole school day) over three week blocks of time. The same experiment was carried out a number of times by several different experimenters in different places. Bloom termed his results "the two sigma problem," because the average tutored student performed two standard deviations above the average normal classroom student. The result, that tutoring works, is quite dramatic.

A Difference in Style Between Novice and Expert Tutors

In this study I show that in some measurable way, our expert tutors engage in a more active-learning tutoring style than a set of novice tutors did. In particular, they force

the student to actively articulate answers more often, using language, as opposed to passively hearing answers.

The CIRCSIM-Tutor project has done several tutoring experiments using novice tutors who had not tutored this topic before, though they may have had tutoring experience. The novice tutors were trained in various ways (trained in tutoring or trained in the problem domain). One of our hopes is that by examining transcripts from these sessions, and by comparing them to the expert-tutor transcripts, we can learn something new about tutoring. Another goal is to answer the question whether expert tutors actually achieve better results. (They do, a result which will be reported elsewhere by others.)

In this session I address the question of who ultimately articulates the fix of a student error: the tutor or the student. This should be a measure of where the tutoring fits on the scale of active to passive.

The bulk of a tutoring session consists of the tutor correcting the student's erroneous predictions one variable at a time. The tutor follows a logical order for those variables, and sometimes other tutorial material is included, but it is always possible to identify conversation segments where the tutor introduces an erroneously predicted variable and the tutor and student discuss the matter until the tutor is satisfied that the student knows (or, to be more precise, the tutor drops the topic). There are three ways the tutor can be satisfied:

- The student says the correct answer, possibly in response to a direct question.
- The tutor says the correct answer.

- The answer has become evident from the discussion.

It is possible that the tutor could drop the topic without one of the three conditions obtaining, but except for a few special instances like running out of time at the end of a tutoring session this case is not often observed. Here is an example of a tutor saying the correct answer (CVP decreases) without forcing the student to say it:

> T: CO also has an effect on another cv variable. Do you know which one this is?
> S: SV
> T: No. SV is one of the determinants of CO, along with HR, but CO does not DIRECTLY affect SV. ... If an increase in CO causes the central venous volume to decrease then the CVP will decrease as well. Does this make sense to you? [N19-36]

And here is an example of another tutor teaching the very same topic, but forcing the student to produce the answer:

> T: ... The heart is taking blood out of the veins faster (CO is up) than the venous return is filling the veins (it takes one minute to catch up) so the volume remaining is decreasing. What happens to the pressure in the central veins then?
> S: The pressure is decreasing.
> T: So, if CO is made to change (say it increases as happened here), what happens to CVP?
> S: It decreases. [K48-74]

The question is: what are the differences between novice and expert tutoring behavior in this regard.

I examined the transcripts of 28 pacemaker tutoring sessions considering only the DR phase of tutoring. I picked the DR phase because hardly any tutoring sessions run into time trouble before this phase of tutoring is complete, and also because there are no ambiguous predictions. Half the transcripts were taught by the experts and half by the novices.

The novice transcripts were all from the April 1996 experiments. This was because of the perception that the novice tutors in April ’96 were better prepared than their predecessors.

For the 14 expert transcripts I took all seven pacemaker transcripts since the advent of our current tutoring protocol (which is what the novice tutors used), and seven from the more recent ones before that. I excluded the transcripts from an experiment where the students were being tutored for the second time around, making them less comparable to the students used in the other sessions.

There were four novice tutors, three of them contributing four sessions each and one contributing two sessions. Among the expert sessions, four were taught by Dr. Rovick and ten by Dr. Michael.

I recorded every prediction table parameter the student made an error on, with the exception of the error of not picking Heart Rate as the primary variable. This error was sometimes ignored by the tutors, I surmise on the grounds that fixing it was an artifact of the tutoring protocol and not necessary for understanding the material.

Most errors were incorrect predictions in the original prediction table. A few were correctly predicted initially, but during the dialogue it developed that the student made a mistake or exhibited a misconception about a variable which occasioned remediation.

I tabulated who first articulated the corrected prediction.

There are four possibilities that I recorded: the student introduced the corrected value, the tutor did so, it is ambiguous or hard to determine, and the error was never fixed. The ambiguous case arises when dialogue proceeds to the next variable after some

discussion, after it seems that the value should be obvious, but the value is not explicitly stated.

I also recorded whether the original error was in the initial set of predictions or developed during the conversation.

It is interesting to note that in fourteen DR sections, using the same pacemaker procedure, there were 39 erroneous predictions in the novice tutor transcripts and 41 in the expert tutor transcripts. This indicates that the students probably were comparable (on average) in their ability to solve the problem.

The net result of all this counting is quite clear: Expert tutors were more likely than novice tutors to have the student introduce the correction: 85% of the time vs. 56%. Conversely, novice tutors are much more likely than experts to introduce the correction: 38% of the time vs. 7%, as shown in Table 1.

Computing $\chi^2 = 11.255$ for this table, with three degrees of freedom, allows us to conclude that the experts and novices are different at a significance level of 0.02. If we discard the five ambiguous and uncorrected cases, we have $\chi^2 = 10.954$ with one degree of freedom, allowing us to say experts and novices are different at a significance level of 0.001.

The implication for CIRCSIM-Tutor is unambiguous. We prefer that when the computer tutor is engaging in dialogue to fix incorrect predictions, it should not *inform* the student of the correct answer if it can *elicit* the corrected prediction from the student instead.

Table 1. Who Ultimately Corrects the Error

	Novice Instances	Novice Percent	Expert Instances	Expert Percent
Student corrected	22	56%	35	85%
Tutor Corrected	15	38%	3	7%
Ambiguous	1	3%	2	5%
Not corrected	1	3%	1	2%
Total cases	39		41	

Recent Developments in Similar Intelligent Tutoring Systems

The proceedings of the most recent ITS conference [Goettl et al. 1998] includes a number of recent developments which show that free-text student input is being worked on in various ways. The PACT Algebra I tutor [Mark et al. 1998] and Algebra II tutor [Corbett et al. 1998] take free-text equation input, as does the Andes physics tutor [Gertner 1998]. The 1998 ITS conference also contains a description of what seems to be the only large modern tutor-directed intelligent tutoring system which accepts free-text input almost exclusively: AutoTutor [Wiemer-Hastings et al. 1998], which will be described in Chapter 7.

Gertner [1998] mentions student frustration with the Andes tutor marking equations as incorrect when the students thought they should be correct. (Sometimes the equations were actually incorrect, but Andes did not explain that very well.) Generally Andes parses equations from student input and compares them to the desired answers. When Andes rejects an equation, Gertner's modifications to Andes come into play. She has a corpus of more than 500 different student-written equations from Andes log files,

enabling her to categorize the phenomena which were causing difficulties. She describes her equation matching algorithm which enables Andes to compare the errant equation to various samples of equations with known difficulties. Different difficulties are believed to be diagnostic of different types of misunderstanding, which enables Andes to select the appropriate hint or instruction.

What is striking to me is the similarity of Andes's equation issues to CIRCSIM-Tutor's issues with free-text input. By having students produce their own equations, you are forcing them to use their recall memory instead of their recognition memory. This causes problems because the students can (and do) type a wider variety of inputs for the tutor to decipher. As in CIRCSIM-Tutor, Gertner has turned some of those unexpected incorrect student inputs into teachable moments, thereby mining even more advantage from the use of free-text student input.

Also reported at the 1998 Intelligent Tutoring Systems conference, Aleven et al. [1998] are moving the PACT Geometry tutor in the direction of accepting free-text explanations for proof steps. What motivates this change is an observation that students seem to be learning how to work the geometry problems in what the authors call a "shallow" manner, e.g. by analogy to previous problems. According to the ACT theory which informs all the PACT tutors [Anderson 1993; Anderson et al. 1995], the knowledge the tutor is trying to teach is divided into declarative knowledge and procedural production rules, as follows:

> An example of a declarative structure in the domain of geometry might be the side-angle-side theorem: "If two sides and the included angles of two triangles are congruent, then the triangles are congruent." Procedural knowledge might

> involve skills of placing triangles into correspondence, determining what an included angle is, setting subgoals, and making inferences. It might also include some frequently encountered uses of this rule such as recognizing triangles as congruent which meet this condition. [Anderson et al. 1995]

The reported difficulty is that instead of learning procedural steps such as "compare these angles because we want to establish a certain prerequisite of the side-angle-side theorem" the students are instead sometimes learning heuristics such as "compare these angles because that's what we did in the other problem which resembles this one" and "if two angles look the same in a diagram then they have the same measure."

One solution being adopted by the Geometry Tutor team is to force the students to give justifications for the procedural steps in their proofs. They hypothesize that the verbal encoding of the rules will cause the students to focus on and learn the logical features of the proof, not heuristics based on visual similarity. Right now the student picks a reason from a list of possible reasons, and the authors of the tutor are working on providing free-text input in the future.

Aside from the observation that forcing students to explain themselves probably improves learning, the implications for CIRCSIM-Tutor are not clear. The explanations of steps in the geometry proofs are short, but the examples of these explanations produced by students on paper-and-pencil tests show that machine understanding of students' explanations will be a messy task. There would seem to be a similarity between proving a theorem, such as occurs in the geometry domain, and causal reasoning, such as occurs in CIRCSIM-Tutor's domain. But we don't observe proof-like verbal behavior in the CIRCSIM-

Tutor project transcripts, where students are required to justify their predictions as a sequence of logical causal steps.

CHAPTER IV

OTHER DIALOGUE-BASED INTELLIGENT TUTORING SYSTEMS

SCHOLAR: Early Intelligent Tutor with Mixed Dialogue

SCHOLAR [Carbonell 1970], an early tutoring system, used language-based tutoring where the tutor often had control. SCHOLAR is often taken to be the first use of artificial intelligence in a tutoring system. It was fundamentally a question-answering system. It contained a semantic network concerned with the geography of South America, of which the student could make inquiries. The semantic network and associated reasoning capability were the basis of Carbonell's claim to have created an intelligent system. It was quite good at locating and printing relevant bits of knowledge. Carbonell also included a feature where the program could ask questions of the student, so that the program and the student would be asking questions of each other in the conversation. SCHOLAR had no dialogue plan and only limited tutorial plans. It had a list of topics to cover, each of which could be randomly realized as a yes/no, multiple-choice, or short-answer question. If the student interrupted a question by asking a question, SCHOLAR could later return back to the interrupted topic. It reacted in different ways to certain variations in the student's answer, such as commenting on spelling errors. But there was no possibility for Socratic tutoring because it had no multi-turn dialogue acts.

Creanimate: Combining Socratic Tutoring with Case-Based Techniques

An interesting relatively recent model for student interaction with an intelligent tutoring system is the "case-based teaching architecture" developed at the Institute of the Learning Sciences, one example of which is the Creanimate system [Edelson 1996]. This

tutor is for teaching animal adaptation: the relation between the morphology of an animal, what it can do, and how it survives. The basic tutorial activity is that students propose modifications to animals. The tutor maintains control of the conversation, asking questions, instructing as needed, and illustrating with cases, often video clips. For example, a student wanted to create a monkey with wings. After offering to show an example of a mammal with wings (a fox bat), Creanimate continues:

T: If your monkey is going to have wings, that should help it do something. Why would you like your monkey to have wings?
So it can ...
S: fly away from its enemies.

It then offers to show some examples of flying. If the student had given a reason which was incompatible with wings, the system would try to teach something about that or offer another suggestion.

The point to notice here is that even though the tutor has control it is offering the students the opportunity to make written suggestions, the results of which alter the teaching agenda. This type of dialogue is fill-in-the-blank, as shown above where the system emitted "So it can ...," leaving the student to contribute "fly away from its enemies." Often the student is presented with a choice of several sentences to complete in this manner.

However the system recognizes only some limited number of different ideas which would fill the blank in "so it can ...." The system extracts what it can use and ignores the rest. One of Edelson's goals is to keep the student engaged, and he evaluates his system partly on that basis. He notes that the conversation is rather inflexible and somewhat repetitive. Also he writes:

> Although the question-and answer dialogues that the Creanimate dialogue manager is capable of conducting provide for rich learning opportunities and a great deal of student control, they fall short of the truly engaging active learning that is the goal of the case-based teaching architecture. Although they give a learner room to propose hypotheses and give direction to an interaction, they place the student in a reactive, overly passive position. [Edelson 1996]

What is interesting is that Creanimate takes free text at all. It is a screen-oriented application, with most of the student input options expressed as buttons and other screen widgets. When the system doesn't recognize an answer, it displays a list of items it will accept. No diagnostic information is derived from deviant student answers. Clearly Creanimate would have been easier to build without free-text input. Considered this way, it becomes apparent that Edelson incorporated free-text input as a mechanism to keep his students engaged and to promote active learning. (Active learning is a topic he discusses at some length.) Although his suggestions for improvement involve such devices as hands-on construction of animated animals, the fact that his system can keep fourth-graders' attention spans for quite some time (he mentions up to two hours) is a tribute to its capabilities.

Dialogue-Enhanced Explanation Systems

Two systems which in some ways have informed the CIRCSIM-Tutor enterprise are the Program Enhancement Advisor (PEA) from Moore [1995] and the EDGE system from Cawsey [1992]. What these systems have in common is that they approach the act of generating explanations as dialogue. It is not enough for the computer to merely generate a sentence or a paragraph describing something it knows. When human explainers

(including human tutors) give explanations, their students often ask questions and influence the explanation being generated.

The PEA system [Moore 1995] advises its users on how to improve their Lisp code. PEA asks the user what aspect of the Lisp function needs improving, then makes recommendations. The user can accept or reject the recommendation, or can ask for explanations. During the explanation, the user can ask follow-up questions.

The maintenance of dialogue is a primary goal of PEA. In order to maintain more natural-sounding dialogue, and especially to be more responsive, Moore maintains both a user model and a dialogue history. She gives examples where the user asks "why?" (there are a finite number of questions the user can ask) and PEA figures out which possible "why" to address and how to address it. This achievement depends on both the user model, which holds PEA's belief about what the user already knows, and the dialogue history, which knows what topics are under discussion and which have already been addressed. A result is that if the user asks "why?" several times in a row different utterances come out, because with each explanation the user model and dialogue history are changing.

The EDGE system [Cawsey 1992] generates interactive explanations of how certain electrical circuits work. In Cawsey's computer model of explanation behavior generating dialogue is primary, and organizing the content is secondary. The system has as its high level goal the issuance of four kinds of dialogue turns: "initial," "explanation," "follow-up questions" (to the user), and "response" (to questions from the user). In a

manner similar to a good many dialogue systems (including CIRCSIM-Tutor) EDGE divides is intelligence up into content-planning and dialogue-planning sectors.

Cawsey started with transcripts of human dialogues of experts describing these circuits to novices. The questions were not all from the novices eliciting better explanations; some of the questions involved the experts quizzing the novices. This is why EDGE includes both follow-up questions from the machine to the user as well as questions from the user to the machine.

It is this observation, that generating explanations requires two-way dialogue, that persuades me there are significant similarities between good explanation systems and intelligent tutoring systems. In CIRCSIM-Tutor the machine has firm control over the agenda, in the explanation systems it is the user who has control. Even given that difference, however, there are resemblance's as follows: 1) the primary planning goal is to maintain a dialogue, 2) at least some of the machine turns during that dialogue are explanations, 3) what to explain and how to explain it are derived, in a large part, from state information (user model, dialogue history) which results from back-and-forth dialogue.

CHAPTER V

PHENOMENA ENCOUNTERED BY THE INPUT UNDERSTANDER

In this chapter I consider some of the varieties of student input which the CIRCSIM-Tutor input understander encounters and might be expected to encounter in the future. The examples of student language come from (or were inspired by) both logs of CIRCSIM-Tutor v. 2 and the transcripts of human tutoring sessions.

An issue here is that CIRCSIM-Tutor is a rather limited conversationalist. We compensate for its less-than-human capability by having the computer keep control of the conversation, ending everything it says with a question or an instruction for the student. The student responds to the questions, and doesn't have an opportunity to get many words in edgewise. The student can type anything, but in practice the students tend to stick to answering the questions fairly concisely, because that is all the machine tutor can respond to.

So in order to find examples of the kind of student input which the machine tutor might encounter in the future, as its capabilities expand, they are taken from the human tutoring transcripts.

Along with the various categories of student language, I frequently discuss how they might be useful to the machine tutor, and how they might be processed by the input understander program.

Note that human tutors frequently ask questions which admit of a broader range of answers than the questions described here. Hume [1995] has documented our tutors asking diagnostic questions designed to elicit explanations from the student. Later in

Chapter 7 I will give examples of such questions and describe how I think CIRCSIM-Tutor might be able to incorporate them. It is also possible in most conversations with human tutors for the students to take, even if briefly, the conversational initiative. I will touch upon this behavior later in this chapter. The remainder of this chapter is devoted to the phenomena encountered in the answers to more ordinary questions. Much of this material was originally documented in [Glass and Evens 1996, Glass 1997].

Even within the domain of fairly simple questions, there is need for the input understander to handle a number of phenomena beyond correct one-word or one-phrase answers illustrated below. The increased demand on the input understander comes from both the variety of student behaviors which we are interested in handling and the increased capabilities of the tutorial and language planners. Note that increases in the input understander's capability must often be matched by increases in the tutorial planner's capability; recognizing and responding intelligently to the new phenomena go hand-in-hand.

Qualitative Changes

Perhaps the simplest question asked by CST v. 2 is "What is the correct value of <var>?" The question is asking for a qualitative change: up, down, or no change.

Usually the student gives a short, fragmentary answer. There is often little syntactic resemblance between the question and the answer. Table 2 contains some responses to the question "What is the correct value of <var>?" What is interesting about this table is first, its sheer variety, and second, the extent to which the answers don't syntactically match the question.

Table 2. Short Answers to “What is the Value of ....”

Word	Comment
up	(adverb)
increase	(verb)
increases	
increased	(predicate adjective or past participle)
i	(drastic but common abbreviation)
unchanged	
no change	
goes up	(phrasal verb)
went up	
it goes up	(a whole sentence)
negative	(adjective)
+	(symbol)
zero	
o	(letter “o” substituting for digit zero)
remains same	(curious grammar)

Yanofsky [1978], in her study of bare noun phrase utterances, had a similar observation, to wit: there is no reason to believe that many short utterances have a syntactically compatible linguistic controller somewhere. This contradicts earlier theories that utterances smaller than a full proposition are derived by deletion from a bigger proposition. The deletion can be controlled either syntactically from a preceding sentence (“Where are you going?” / “Home.”) or pragmatically (during a chess game: “Your move.”). In these instances the short utterance can be seen to fit as a constituent of a larger, but unexpressed, proposition, to wit: “Home” is an adverbial modifier or a location object in the sentence “I am going home” and “Your move” is the object of “It is your move.”

Table 3. Responses to "Where are you going?"

	Answer	Meaning
	Home	"I am going home."
*	To home	* "I am going to home."
	To my home	"I am going to my home."
	Away	"I am going away."
*	From the enemy	* "I am going from the enemy."
	Away from the enemy	"I am going away from the enemy."

If this kind of argument is unfamiliar to you, let us consider possible answers to "Where are you going" to see how it works in Table 3. Unacceptable answers are marked with an asterisk.

You can see that to a first approximation, a short answer must syntactically fit the question, in this case it must replace the constituent X in "you are going X." Yanofsky's catalog of NP utterances proved that this is not always true. She gives the following as but one of many examples. Suppose that Bob and Carol are walking down a busy street and Carol suddenly blanches:

Bob: What happened?
Carol: My ex-husband.

Clearly there is no linguistic controller for "my ex-husband," as it is generally not correct to say "my ex-husband happened."

Frederking [1988] has a nice computational model of ellipsis and pronoun resolution called Psli3. In a sequence of utterances in a dialogue he constrains the antecedents of ellipses partly by their discourse functions, as opposed to by their syntax. Here are two examples of bare noun phrases which his program can resolve [Frederking 1988, pp. 32–33]:

Hand me a hammer. A wood-handled one.

Show me your hands. The other side.

Frederking's program depends heavily on case frames for resolving semantic and pragmatic constraints and finding antecedents. Once an utterance has been parsed it can be matched against the instantiated case frames produced by previous utterances, regardless of whether or not there is a syntactic match.

And that is indeed our experience as shown in Table 2. Answers such as "increase" and "no change" do not fit neatly as a constituent of any plausible syntactic explanation of "What is the correct value of <var>?", but they make perfect semantic sense.

Occasionally a student will state an answer as a complete proposition, for instance "cardiac output increases" when just "increases" would be adequate. We have to produce a logical form for such a sentence and then match it to the question. Consider the hypothetical exchange:

T: What happens to stroke volume?
S: Cardiac output increases

Clearly the input understander can't simply notice the word "increases" and ignore the rest. Another question is how to handle the following hypothetical case:

T: What happens to stroke volume?
S: It, like entropy, increases with time.

I claim that in this case the best response is for the input understander to notice "increases" if it cannot recognize the rest of the sentence.

There is no profit in parsing many of these answers with a grammar of complete sentences. Indeed, following both Yanofsky and Frederking, there is no basis for

considering them as reduced sentences at all. Instead, in the CST input understander lexical entries for observed short-answer words and phrases contain pointers to the relevant logical concept. The above examples cover only three concepts: up, down, and nochange. It is the job of the input understander to try to match these concepts, taken from the lexical entries, to the question being answered, which is available from the planner in logical form.

In our keyboard-to-keyboard transcripts as well as logs from the machine tutor we have fairly extensive examples of these kinds of answers. Despite the wide syntactic variety, there really are only a limited number of them, so we can catalog them all.

There is a fourth concept which is needed for these questions, change, meaning up or down without specification. It is not attested as a student response in any CIRCSIM-Tutor log that I have seen, but can occur as a result of incomplete understanding of a student's answer such as "it doesn't change." This might happen, e.g., because of a typing error. More often, the change concept occurs in the tutor's utterance, for example when the tutor gives a negative acknowledgment as in the following hypothetical exchange:

T: What is the value of Mean Arterial Pressure?
S: Unchanged
T: Wrong, the value of Mean Arterial Pressure must be changed from its initial value.

Phrasal Answers

Less fixed, because students exhibit linguistic creativity, are certain phrasal answers. In response to the same question "by what mechanism is <var> controlled?" students have been observed uttering "neural," "nervous system," "parasympathetic

nervous system," "sympathetics," "sympathetic stimulation," "sympathetic tone" and "reflex" among other answers. Given such data, we would not be surprised to see "neural stimulation" or "reflex system."

Just as in the short answers, the lexicon yields up a concept associated with many of the words and phrases. In this case, the relevant concept is nervous system and that concept is in the lexical entries for the adjective "sympathetic," the noun "reflex," the adverb "neurally," and so on.

Note that the eventual interpretation of a noun phrase does not necessarily come from the head noun. For example, "system" as the head noun in "neural system" contributes nothing; it is "neural" that carries the meaning. This observation is one factor which made Lexical Functional Grammar awkward. Generally in LFG interpretations the head word provides the basic meaning in the functional representation of a constituent and the meanings of modifiers are included in other ways. The formalism makes this structure very convenient, but it is not always what you want.

Flagging all neural words as "nervous system" in the lexicon is too simple-minded. Consider the difference in the following two extracts from the human tutoring transcripts. The significant parts in the second extract are marked:

T: Can you tell me how TPR is controlled?
S: Autonomic nervous system.
T: Yes. And the predictions that you are making are for the period before any neural changes take place. [K10-29]

T: How is TPR controlled?
S: Sympathetic vasoconstriction.
T: Right, TPR is primarily under neural control. We're talking about what happens before there are any neural changes. [K11-49]

The student's answer in K10, "autonomic nervous system," exactly matched the tutor's intention, which was to talk about neural changes. The student in transcript K11 gave an answer "sympathetic vasoconstriction" which, while correct, was slightly farther from the desired answer. The tutor reflected back a corrected version: "TPR is primarily under neural control." The tutors do not always exhibit this behavior, often assuming that an answer such as in K11 is close enough that it does not need the implicit correction, but it occurs frequently enough that we should consider supporting it in the machine tutor. There also occur instances in the transcripts where some of the semantic differences among the various neural answers are important (for example the difference between parasympathetic and sympathetic nervous systems) although we have not formalized those tutoring strategies yet so they do not occur in CST's conversation.

In CST v. 2 all neural answers are treated alike. The tutor always responds by reflecting "Correct, TPR is controlled by the nervous system," no matter which neural answer the student typed. It is a fail-safe response, suitable for answers which were both exactly and approximately correct. However, in order to make finer distinctions in the future it is necessary to put these distinctions in the lexicon. One word, e.g. "sympathetic," might sometimes be equivalent to "neural" in response to one question or equivalent to "autonomic" in response to a different question or even specifically "sympathetic" in response to a third question. The solution is to have a little ontology containing all neural terms. In this ontology "sympathetic" and "parasympathetic" are both part of "autonomic", which is part of "nervous system", which is a "mechanism." We put the specific meaning of the term in the lexicon. For purposes of answering

Table 4. Some Attested Spelling Errors

Text fragment containing unknown input word	Replacement from lexicon	What happened?
I didn't think that radius would chande (K51-39)	change	substituted one letter
the radiuis of the arterioles (K51-80)	radius	interpolated one letter
how stretched the msucle fibers are (K50-32)	muscle	transposed two letters
Whyndid you predict that (K49-50)	Why did	space character mistyped
to prdict a change in sv (K48-262)	predict	elided one letter
doesn't sound to positive (K47-181)	too	wrong word, won't parse
think of the eq CO=HR x SV (K47-141)	equation	drastic abbreviation

particular queries the ontology tells us whether the meaning of the given word fits the desired category.

Spelling Errors

Spelling errors of various sorts are a striking feature of the tutoring transcripts. Chapter 8 contains a discussion of the spelling corrector and how it is integrated into the new CST v. 2.5 input understander.

From a computational point of view, the term "spelling error" is being stretched here to cover words which occur in the text, do not occur in the lexicon or cause parse failure, but could be replaced by a word or words from the lexicon to achieve the intended effect. Several phenomena can be so described. There are some examples in Table 4.

To get a flavor of what happens in actual input, in the following sentence we can find two spelling errors, a missing letter and a transposition:

EDV deterines how stretched the msucle fibers are. [K50-tu-32-2]

In addition to spelling problems, one can observe many impromptu abbreviations in the transcripts. When conversing with human tutors, students often abbreviate with abandon. “Inotropic state” can become “inotropic s.” “Parasympathetic” can become “parasymp” or “para.” There is a stock of standard abbreviations which can be used at any time. “Cardiac output” is usually typed “co” or sometimes “c.o.” The standard abbreviations are in the lexicon, and are thus not subject to spelling correction. But the impromptu abbreviations are not in the lexicon, so it is the spelling corrector which ultimately handles them.

In the course of rewriting the program which numbers transcripts I used some rules which help with the impromptu abbreviation problem. These rules have not been incorporated into the CST input understander, but they can be included in the future if a better handling of impromptu abbreviations is needed.

The input to the numbering program is a raw transcript, where the turns are separate blocks of text. A primary job of the numbering program is to separate out the sentences in each turn and number them within turns. Usually whoever types a multi-sentence turn puts periods at the ends of sentences (they are frequently omitted in single-sentence and fragmentary turns), but the periods which occur in abbreviations can cause difficulty. A simple algorithm to collect interstitial and final periods and blanks from strings of single letters works rather well, so that if someone types “a.n.s,” “a. n. s,” “a.n.s.,” or “a. n. s.” it will not cause extra, short, sentences. Some of the period-delimited abbreviations are in fixed expressions, such as the “dr.” in “Dr. Michael” and “Dr.

Rovick" (you can't count on capitalization). I collected from the transcripts I was numbering a table of known period-containing abbreviations. Any abbreviation found in the table is presumed not to end a sentence. Note however that it is insufficient to simply put "dr." in the table as an abbreviation, since many sentences will legitimately end with "dr." meaning "direct response." The table must also contain the doctor's name in order to distinguish the title "dr." from the stage "dr." Note also that some students misspell their professors' names, so some care is required.

The program works better than its predecessor, and was used for numbering most of the thirty novice tutoring transcripts. It is not perfect however; hand-editing of the result is required. Typically about 2% of the turns need to be renumbered by hand.

Hedges

Hedged answers occur frequently when students are conversing with human tutors. "How about cardiac output?" is one attested answer to a question [K25-st-53-1]. We see hedged student utterances frequently in the transcripts of human tutoring sessions, and even occasionally in the logs of the machine tutor. It might be possible to make use of hedges in the student model and perhaps in dialogue generation. A heavily hedged response, for example, might deserve an explicit positive acknowledgment, even though positive acknowledgments are not always given explicitly.

CIRCSIM-Tutor v. 2.5 does not have any mechanism for handling hedges, so they remain largely unimplemented in the new input understander. Right now, in the new input understander only the question mark signals a hedge, and this information is not reported to the planner.

It is hard to discern a common principle which would allow the input understander to recognize hedges. I have examples of hedge phrases from our transcripts (including the popular question mark), which the input understander converts to a hedge token in the input. It makes no practical sense to parse "I think cardiac output increases" into a main verb "think" with a complement sentence. Consider first that few real answers to the machine tutor's questions are complicated enough to contain an embedded sentence, and second that the student may also utter "cardiac output, I think." Therefore I would include "I think" as a phrase in the lexicon, flagging a hedge. The word "probably" receives similar treatment, along with "I guess," "possibly," "what about," and other common hedge phrases.

One can catalog any number of hedges by looking at the transcripts. It seems certain that students will always be able to create new ways to hedge, but absent a general principle to recognize hedges cataloging known ones is what we will do.

Having said hedges are largely recognizable by key words and phrases, it must be noted that sometimes there is a syntactic clue: students do sometimes hedge by simply asking a question, perhaps with no punctuation at all. So identifying the inverted verb (if the student entered a sentence) or finding a *wh-* word may be useful for identifying answers hedged in this fashion.

Related to hedges are phatics and expressions of frustration. Phatics are the small meaningless phrases used to establish social communion. Seu [1992, p. 46] claims to have found and counted such phrases in the transcripts, expressed as words such as "OK", "well," and "oh." It is not clear to me whether students will attempt to achieve social

communion with a computer tutor, but perhaps we can be ready if they reflexively try. We do not see expressions of frustration (which Seu classifies as a form of phatic) in the transcripts, possibly because the tutor is usually the student's professor. In the past I have seen them in logs of library patrons using an on-line catalog [Glass 1986]. In the most recent (November, 1998) logs of a physiology class using CIRCSIM-Tutor at Rush Medical College there are two attested examples.

Category Errors

A certain number of student answers are of the wrong category, not what was expected. Some of them are of the "kitchen sink" variety, such as this one (marked with an asterisk) from the November 1998 sessions at Rush Medical College:

T: By what mechanism is Total Peripheral Resistance controlled?
* S: Metabolic factors
T: Is the mechanism of control neural or physical?
S: Neural

In this case the input understander didn't recognize the student's input, as it is beyond the normal semantic range used in building the lexicon. The response "is the mechanism of control neural or physical?" is an error message from the input understander. In the past the input understander emitted the error message "I'm sorry I don't understand you, please rephrase" in response to all problems. Experience has shown us it is important for the input understander to indicate the category of answer it expects. All the new input understander error messages follow that pattern.

Perfectly recognizable input can still be a category error. Suppose the student in the example above had typed "increase," viz.:

T: By what mechanism is Total Peripheral Resistance controlled?
S: Increase
T: Is the mechanism of control neural or physical?

This kind of response happens with some frequency.

In CIRCSIM-Tutor v. 2 there is no good way to report such a response to the discourse planner, and the discourse planner couldn't use such an answer. So it is the responsibility of the input understander to screen the category of the given answer against the categories of acceptable answer and issue an appropriate response to the student.

Near-Misses and Other Appropriate but Unexpected Answers

We have observed a number of categories of student answers which aren't what the tutor expected, but aren't incorrect either [Glass 1997]. Here is an example, with the student's unexpected response marked with an asterisk:

T: What is the primary mechanism of control of TPR?
* S: Radius of arterioles.
T: Yes. And what is the primary mechanism by which arteriolar radius is controlled?
S: Sympathetics.
T: Yes. [K12-37]

Here the tutor wants to make the point that TPR is controlled by a nervous system reflex. The nervous system happens to do that by varying arteriolar radius. Thus arteriolar radius is not a wrong answer; it really does control TPR. The student should not be contradicted. The student's answer doesn't serve the immediate tutoring goal either. If an answer can be made to serve, by having the tutor leading the student from the unexpected answer to the desired answer, we call it a "near-miss."

There are many similar cases involving other concepts in our transcripts. For instance, if the tutor asks "what determines stroke volume" the student might reply

"ventricular filling." Again, this answer is correct but it doesn't serve the immediate tutoring goal. The tutor in that case may try to lead the student from ventricular filling back to the desired answer, which is central venous pressure or, in older transcripts, right atrial pressure. Here is an attested example:

T: What are the parameters that determine the value of SV?
S: Filling and contractility?
T: Right, but which parameter in the table reflects filling?
S: RAP [K44-116]

For processing these kinds of answers the input understander needs to make some sort of preliminary relevance judgment. If the question is about the determinant of a parameter, then any parameter is in the right category of answer. However when the answer to the question about what determines TPR is "arteriolar radius" we discover that the category of desired answer ("reflex") is a neural control mechanism while the category of the actual answer "arteriolar radius" is a measurable parameter. It would appear that "arteriolar radius" cannot be an answer to the tutor's question, as it isn't the same type of object that is being asked for.

In order to identify a near-miss answer it is necessary to know the current tutoring goal. In CIRCSIM-Tutor v. 2 this information is not available to the input understander. The discrimination among types of student answers is performed in the student modeler. However there are only a finite number of possible near-miss cases which the student modeler and planner can recognize, so the input understander can be aware of the various types of acceptable near-miss answer categories.

How the tutorial planner handles the near miss is not an input understander question. If it has the appropriate schemas, it may be able to follow up on the near miss,

and then return to its tutorial plan in progress. The point here is that if the planner can engage the student's partly-correct idea, this is more responsive than simply treating the near miss as an incorrect answer. It can be akin to a student initiative, in the sense that the planner suspends its current plan, deals with a student idea, then returns to normal tutoring.

We like near-misses because they are implementable, they are something that students demonstrably do, and it is a step toward making a planner which adapts its tutoring plans in a cooperative response to the student's input.

Misconceptions are another type of student answer which are appropriate but unexpected. This is an example of a common misconception described by Sanders [1995, p. 97]:

T: In what way is cardiac contractility CC controlled?
S: It's controlled by the volume of blood in the compartment and affected by ionotropic changes.
T: Not quite. Changing the volume changes the performance of the muscle via the length/tension relationship, i.e. Starling's Law.... [K31-60]

In this case "volume of blood in the compartment," a parameter, is given as a determinant of inotropic state. Again, this appears to be a category error. What was expected was a neural mechanism of control, what was received was a parameter. From the input understander point of view it is the same as a near miss.

However this answer is indicative of a common misconception confusing inotropic state with a phenomenon known as the Frank-Starling effect. From the tutorial planner point of view it triggers a diversion which remediates the misconception.

Notice that the misconception is not understood and diagnosed by using any deep model. Instead the misconception was cataloged by examining the tutoring transcripts. CST recognizes certain short answers as being diagnostic of certain misconceptions.

Misconceptions and near misses are powerful examples of why it is useful to allow free-text student input. The above examples are comprised of very short, easily understood answers to closed questions. Nevertheless they provide teachable moments. If the students had no option for free text input, but instead had to pick from a menu of answers, the right answers would always be staring them in the face.

Zhou et al. [1999] detailed CIRCSIM-Tutor's most complete typology of unexpected student answers to date along with examples of how the planner might respond to each. She has implemented many of these in the CST v. 2.5 planner. All involve typical short student answers such as described in this chapter.

Student Initiatives

Student initiatives are utterances "by which the student is apparently trying to modify the course of the tutorial dialogue and could reasonably expect to do so" [Sanders 1995, p. 59].

One student initiative attested in logs of students using CST v. 2 is "I don't know," sometimes taking the form "dunno" and sometimes expressed in more complicated ways. The input understander handles this case. CST responds simply with the answer to the question and proceeds.

A discussion of initiatives is too big to include here, but it is useful to see a sample initiative to become acquainted with the nature of the problem as an input understanding

issue. Here is a student initiative involving simple language and concepts close to the short answers we have seen so far in logs of the machine tutor:

T: ...what about the rate at which blood is being removed from the central blood compartment?
S: The rate would increase, perhaps increasing RAP??? [K3-54]

In this example the underlined part is a student initiative. The hedged prediction about the change in Right Atrial Pressure is extraneous; it does not respond to the question at hand, which asked about the rate of change of Central Blood Volume. In fact the context for this exchange was an attempt to remediate an incorrect prediction for RAP, so it seems the student might have been trying to skip ahead to the conclusion.

Even this simple example presents a messy problem in input understanding. One issue is that there are two clauses, of which one is responsive to the question and the other one isn't. So it would be necessary to decode that there are two clauses and extract the responsive one. Next there may be a question of causality. The student's response can be paraphrased as three independent statements: "the rate increases," "the increasing rate causes an increase in RAP," and "RAP increases." Deciphering this requires some work. Finally there is the issue of divining the student's intention. In the above case the student's intention cannot be known except by referring to the tutorial context. As a practical matter, the student's intention would have to be categorized into one of a small number of types on the list of types the planner has plans for.

In the CIRCSIM-Tutor project student initiatives in the transcripts have been extensively cataloged by Sanders [1995] and Shah [1997]. Shah's classification is too complex to describe here, as it is composed of four somewhat independent components,

but her list of communicative goals should suffice to illustrate the range of attested initiative types:

- Request for information
- Request for confirmation (the student proposes a notion)
- Support (the student echoes back a notion received from the tutor)
- Challenge (the student appears to disagree with the tutor)
- Conversational repair
- Refusal to answer
- Time delay
- Other

In this thesis I do not discuss the question of handling initiatives in general. However there are a few simple initiatives which are attested and are relatively easy to recognize (such as the "don't know" initiative), and that are handled in the new input understander described in Chapter 8.

Equations

There are two equations from the CIRCSIM-Tutor problem domain which frequently occur in the transcripts: CO = HR x SV and MAP = CO x TPR. These equations occur in various ways in student language.

In the transcripts the tutor occasionally prompts for an equation explicitly, for example like this:

T: Can you write an equation that shows what variables determine MAP? [K40-106]

CIRCSIM-Tutor v. 2 currently never asks for an equation. However in the transcripts of human tutoring we sometimes witness students providing equations as parts of their answers, even when an equation was not requested. Can we expect this behavior from students using the computer tutor? Indeed here is an attested example from a log of a paid volunteer medical student using CIRCSIM-Tutor during an experiment in February, 1994:

T: What are the determinants of mean arterial pressure?
S: tpr X sv

We note that the correct answer would have been TPR and CO, not SV, but the point is that it would be useful for the input understander to recognize equations.

There is variety in the syntax of equations expressed in student language, including mixing of symbols and English. A fuller description of the expression and use of equations is in Chapter 9.

Persuading Students to Use Our New Features

It is not surprising that students use much simpler language conversing with the computer as opposed to human tutors. One imagines that students also carry expectations that they have accumulated from previous experiences with question-asking computers. If a computer program asks how many Cabinet officials the United States has, for instance, few people would respond with "about a dozen" or "why do you want to know." Computers do not in general reinforce giving such answers. The fact that CIRCSIM-Tutor asks closed questions reinforces this behavior.

A nice example of what appears to be a person having confused expectations of a computer dialogue is the following, from the November 1998 session at Rush Medical College. The unexpected student answers are marked with an asterisk:

T: What is the determinant of Central Venous Pressure?
(1) S: Blood Volume
T: Blood Volume determines CVP. However, no change in blood volume occurred since DR.
(2) * S: OK
T: Please respond with prediction table parameters
(3) * S: Unchanged
T: Please respond with prediction table parameters
(4) S: Mean Arterial Pressure

In this segment student answer (1) was not expected, but it had some truth in it, so the computer tutor responded with an acknowledgment and a hint. But in this turn the computer did not issue a new question. Instead it silently handed control back to the student. (This behavior was not intended by the programmers.) I cannot know what went on in the student's mind at this point, but the student responded in (2) with "OK," which would be a perfectly reasonable response to a human. It seems to me that in a normal human conversation the mooted question still stands; it would be odd at this point for the tutor to re-ask the question. That the student has lost sight of the question is shown by answer (3). I attribute this lapse to the student's expectation that the computer always issues a question, and does not behave in the manner of normal human conversationalists. The above non-question occurred several times in the November 1998 sessions at Rush. On at least one other occasion it seemed to confuse a student, who typed the same wrong answer several turns in a row.

Because people have prior expectations about how to answer computer-generated questions, we expect that some of the phenomena we hope to capture and utilize won't often occur in nature. For example, we observe that human-tutored students often hedge their answers, turning "increase" into "increase?" and "cardiac output" into "probably

cardiac output." If we want CIRCSIM-Tutor to respond to hedges we may need to give students permission to hedge.

Instead of giving students explicit permission, it might be possible to include some sample tutoring dialogue in the instructions. Finding neutral illustrative dialogue may be a bit tricky, as we do not want to corrupt possible experiments by giving away answers to some of the problems. But illustrating hedged answers with question marks and adverbs, for example, may have more impact on the student than granting explicit permission.

Some of the phenomena we hope to handle, such as spelling errors, occur naturally. We do not need to give students permission to misspell words in order for misspellings to occur. But with human tutors students seem to misspell with abandon, and invent impromptu abbreviations, more often than they do with the machine tutor. It might be nice to increase the student's expectation that the computer is capable of such treatment, but if students receive positive reinforcement when typing mistakes occur, giving permission may not be necessary.

CHAPTER VI

TECHNOLOGICAL BACKGROUND TO THE INPUT UNDERSTANDER

Semantic Grammar in SOPHIE, an Early Free-Text Input ITS

An early rather sophisticated free-text interface for an intelligent tutoring system belonged to SOPHIE, a system to tutor the debugging of power supply circuits [Burton and Brown 1979, Brown and Burton 1975]. The examples that Burton and Brown publish are quite impressive. Here are four sample turns of student input (tutor's responses omitted):

a) What is the bse emitter voltage of the voltage limiting transitor?
b) What about the current limiting transistor?
c) What should it be?
d) What is the current through R22 when it is shorted?

[from Brown and Burton, 1975]

Some of the notable capabilities evinced by the above examples are:

- Correction of "bse" to "base" and "transitor" to "transistor."
- Fairly complex stacked nouns.
- Sentence b) has no verb.
- Identification of "base emitter voltage" as the controller of the ellipsis in b). There are alternative possibilities, e.g. voltage to ground.
- Resolution of "it" in c) to "base emitter voltage of the current limiting transistor."
- Resolution of "it" in d) to "R22," a subordinate constituent embedded in a preceding noun phrase. Contrast with "the current through R22 when it is normal," where "it" would refer to "current through R22."

This by no means covers all of SOPHIE's input understanding capabilities, but it is enough to give you the flavor.

Brown and Burton [1975] explain why they chose to concentrate on the kinds of features illustrated above:

> To compound our problem we discovered from using early versions of SOPHIE that when a person communicates with a logically "intelligent" system he inevitably starts to assume that the system shares his "world-view" or is at least "intelligent" in the linguistic art of following a dialog. In other words, SOPHIE had to cope with problems such as anaphoric references, context-dependent deletions, and ellipses which occur naturally in dialogs. In fact handling these constructs seemed more important than building a system endowed with great syntactic paraphrase capabilities. [Brown and Burton 1975, p. 324]

For SOPHIE Burton and Brown developed what they called a "semantic grammar." The notions behind the semantic grammar are not at all unusual today. It resembles an ordinary context-free grammar for a small subset of English. The non-terminals, however, correspond to the semantic concepts which SOPHIE knows about, such as measurement, transistor and so on. Hypothetically, in order to parse a phrase such as "voltage across capacitor C2" you could have a phrase structure production such as:

<MEASUREMENT> := <MEASURABLE/QUANTITY> <PREP> <PART>

The derivation would then become:

<MEASUREMENT> → <MEASURABLE/QUANTITY> <PREP> <PART>
→* voltage <PREP> <PART>
→* voltage across <PART>
→* voltage across capacitor C2

A drawback to such a scheme is the proliferation of productions as the number of non-terminals increases. In an ordinary grammar, all noun phrases, for example, are

derived from a single non-terminal <NP>. In the semantic grammar, there will be a number of structurally similar rules, one for NPs which refer to measurements, one for NPs which refer to parts, one for NPs which refer to terminals, etc. Because the domain of discourse is restricted, as it is in an intelligent tutoring system, the problem is manageable.

Actual parsing in SOPHIE is accomplished by encoding each production as a Lisp procedure. The result is a top-down recursive descent parser with a separate routine for every production.

The rule routines are able to take care of quite a bit of syntactic sloppiness. The semantic grammar gives good predictions of the types of constituents to expect next, given what has already been parsed. The routines make various corrections to the input text, and skip over unnecessary words, in an effort to find the next predicted constituent.

These parsing routines are also responsible for anaphora resolution and for producing the semantic representation of the sentence.

Simple deletions are resolved by knowing the dependencies between the various concepts. For example, the phrase “the voltage at the collector” is analyzed as a measurement, meaning it has a measurable quantity (“voltage”), and a terminal (“the collector”). However, terminal is further subdivided into a terminal-type and a part, where “collector” matches only terminal-type. In “the voltage at the collector” the part is missing. A look into the map of the dependencies between concepts reveals that the word “collector” implies that some transistor is present. SOPHIE then looks backwards through its discourse history to find the most recently referred-to transistor, which fills in for the missing part.

Ellipses are handled by enumerating their possible forms directly in the grammar. An ellipsis is an utterance which expresses part of a proposition, with the remainder to be picked up from context shared between the speaker and the hearer. Burton and Brown [1979] give these examples, among others, of elliptical utterances which SOPHIE correctly recognizes. The first sentence is complete, but the following three are ellipses, where the omitted matter is marked with ø:

> What is the voltage at Node 5?
> ø At Node 1?
> And ø Node 2?
> What about ø between nodes 7 and 8?

The grammar has productions which can describe utterances such as the above, all derived from the non-terminal symbol <ELLIPSIS>. The routine which handles these productions knows it is resolving instances of ellipsis, and what constituents are missing, so it can examine the discourse history to find the missing referent.

It is important to note that had the elliptical utterances been parsed as sentence fragments or phrases according to a standard phrase structure grammar of English, recovering the deleted constituents might have been more convoluted. The phrase "what about X" requires that X be a noun phrase, but in the above example it is a prepositional phrase. The result of such a parse, if that parse is not informed by the idea that the whole phrase is an ellipsis, is not likely to be useful. (In fact, SOPHIE's grammar seems to recognize "what about" as an introducer of ellipsis.)

There are big differences between SOPHIE and CIRCSIM-Tutor, of course. The primary one is that SOPHIE's conversation is under student control (the student asks the questions, the computer responds), and CIRCSIM-Tutor's is the other way around. This

affects the kind of language the tutoring system sees as input. The language that CIRCSIM-Tutor sees is usually simpler. Because it is asking simple questions of the student, the student rarely has to express a complete proposition. Deleted constituents should almost always refer back to the immediate question. On the other hand, CIRCSIM-Tutor has goals for the conversation. The student's answer may or may not match the tutor's intentions, or may match them in unusual ways. So although CIRCSIM-Tutor's parsing task might be strictly easier than SOPHIE's, its "understanding" task is rather different.

Message Understanding

An active area of natural language processing research known as "message understanding" or (more recently) "information extraction" provides a variety of techniques which can be useful for input understanding in an ITS. The salient characteristics of message understanding are 1) its ability to process unedited text and 2) its ability to extract from the text only the information which is desired.

Message understanding is relevant to the CIRCSIM-Tutor v. 2 input understander task because 1) it is robust with respect to messy text and 2) it usually suffices to extract an answer to the question and ignore the rest of the student's utterance. I have adopted information extraction techniques for the CIRCSIM-Tutor version 2.5 input understander, described in Chapter 8.

The message understanding task starts with real-world text. No editing is allowed. The ground rules for this enterprise were set in the (as best as I can determine) original message understanding system, called NOMAD [Granger 1983]. NOMAD, as well as a

number of succeeding systems built by other people, examined Navy ship-to-shore messages, which looked like this:

a) Challenged ship refused to heave to
b) Locked on open fired destroyed

This is not beautiful English. Message a) may be analyzed as two sentences, both missing their subjects, viz.: "We challenged a ship; it refused to heave to", or as a single sentence missing an article, viz.: "The challenged ship refused to heave to". Message b) is almost certainly three separate sentences glommed together, with large parts missing from all three, and the middle one ("open fired") has a grammatical error to boot.

The output of the message understanding task is a set of filled-in templates. These templates are like forms to fill out. You can imagine you have a set of a dozen or so stylized kinds of incident report forms: e.g. battle with another ship, communication from ship to some other party, change in location, change in status, etc. Then you read a message, decide which incident report forms apply, and fill them out as best you can.

This is why the enterprise has more recently come to be known as "information extraction" [Hirschman and Vilain 1995]. The goal is to fill out those templates by whatever means necessary, extracting the information from the text, without necessarily producing a complete semantic representation of the input text. In many applications, most of the input text does not contribute to the output and is ignored.

Message understanding is being driven by DARPA, which runs a message understanding competition and conference every two years or so [MUC 1993,

MUC 1995, MUC 1998]. Descriptions of the tasks and scoring are available from Science Applications International Corporation, which runs the conferences under contract.

The competitions have grown from six teams at the first one in 1987 to eighteen in 1998. The assigned competitive task has grown from a couple of messages and a couple of different information templates to hundreds of messages and fifty different possible information templates. The task isn't restricted to navy ship-to-shore messages anymore; the last few have used unedited newspaper text from the Wall Street Journal and the New York Times. Teams participate first in a dry run where the tasks are the same as in the eventual competition but the problem domain is different. They do not learn the actual domain until shortly before the competition. For example, in MUC-7 the dry run involved newspaper stories about airplane crashes while the eventual competition involved stories about space launchings.

There are three basic categories of MUC tasks, called "named entity," "coreference," and "information extraction." The named entity task is to locate and categorize all the named entities in a passage. Generally these are proper nouns, which must be categorized into place names, company names, etc. Also included are a great number of entities like dates and times, profits, numbers, and so on. The coreference task is to discover which items in the text are coreferential. The information extraction task involves filling out the templates mentioned above.

To see examples of named entities and coreference, consider the following hypothetical text:

> Ford Motors announced lower than expected second quarter profits today.... The second largest auto company attributed the decline in earnings to....

Here "Ford Motors" is a named entity, coreferential with "the second largest auto company." Similarly, "second quarter profits" is a named entity, coreferential with "earnings."

As one might expect, there is no one method which is used in message understanding. One of the dominant technologies is cascaded finite-state transducers [Roche and Schabes 1997]. However different components of the same system may often use differing technologies.

In order to process student initiatives in future versions of CIRCSIM-Tutor the message understanding paradigm may be useful:

- The tutorial planner (or whatever planner is responding to student initiatives) would have a finite set of initiative categories it could recognize. (In the Chapter 5 section on student initiatives there is a short list.)
- Each category of student initiative in this regime would have a template, describing the information requirements of that initiative.
- The input understander scans the student utterance to see which template (if any) it might be able to fill in. If a match is found the information to fill the template is extracted.
- If no initiative is recognized, the utterance is taken to be an answer to the last question posed by the ITS.

In fact this is exactly the approach I used in the CIRCSIM-Tutor version 2.5 input understander for recognizing and processing the one student initiative currently handled: variations on "I don't know."

This suggestion contrasts with an alternative approach where the input understander tries to build a semantic representation of the student's statement. However much of the information in the student's statement is probably not needed in order to create some sort of intelligent response. Furthermore the student probably has ways of expressing this information which the input understander doesn't know how to parse or represent. Both factors make it much more likely that a semantic representation cannot be built, and input understanding will fail.

One can speculate that for longer, multi-turn, student initiatives the message understanding approach could well fail, due to the need to model the progress of the student's plan across turns. Long, multi-part explanations may also pose difficulties. But it seems to me that for that first big step of handling meaningful but short student initiatives, the message understanding model is useful.

More Recent Approaches to Symbolic Understanding

Two recent pieces of work representing the latest in symbolic approaches toward comprehending messy real-world text are the GLR* parser [Lavie 1996] and the ExtrAns "answer extraction" system [Mollá Aliod et al. 1998]. I include them here because it is instructive to know what are the state-of-the-art techniques among people solving problems similar to input understanding in the non-tutoring-system world.

The GLR* parser is a descendant of LR(0) shift-reduce parsing. It is derived from Tomita's Generalized LR parsing algorithm, a way of extending the LR(0) algorithm to cover the full range of context-free grammars. LR(0) parsers are quite fast, but they can cover only a subset of the context-free grammars. Tomita's algorithm replaced the basic stack with a more complicated data structure, one that holds the results of many alternative parses at once and combines their common parts. While Tomita's algorithm parses it simultaneously advances (shifts and reduces) all alternative derivation paths; the combining of common segments of the alternative derivations is what makes the algorithm efficient.

Lavie was working in speech recognition. Continuous speech can often be characterized as a proposition being expressed plus many interstitial noises and less relevant words. Furthermore speakers engage in frequent repair behaviors, such as backing up a few words and starting over, which can be processed by eliding a few words.

If you are committed to trying to produce a parse of the utterance, one approach is to skip the words which are not within a parsable proposition and parse the rest. So which words might be skipped? You could try selectively eliding different words, in all combinations, and seeing whether the resulting utterance is parsable. The longest parse you achieve this way (the parse which succeeded after dropping the fewest words) can then be taken to be the most interesting. However experimentally parsing every possible subset of the input utterance would be very expensive, since for an n-word utterance there are about $2^n - 1$ possible reduced utterances.

The principle driving the GLR* parser is to extend the Generalized LR algorithm so that it combines (where possible) and processes the alternative parses which result from skipping different subsets of words. In addition to expanding Tomita's parsing algorithm and data structure to make this possible, Lavie added a number of methods (e.g. statistical preferences, restrictions on the number of skipped words in a row, reorderings of the search space) to reduce the number of alternatives considered and increase the speed.

GLR*'s robustness comes from its ability to try to skip many alternative subsets of words and still operate with computationally tractable bounds. This is particularly necessary in any system which needs to be interactive. It also benefits from a fairly comprehensive grammar, since it is trying to produce a parse of the utterance. Ultimately it produces a functional structure, similar to the f-structures of LFG. Although earlier versions of CIRCSIM-Tutor's input understander had this kind of comprehensive parse as a goal, that step has not proven to be necessary for CST input understanding.

The ExtrAns system [Mollá Aliod et al. 1998] is an attempt to build something which is more specific than information retrieval yet covering a broader, shallower domain than a question-answering system. It answers questions about Unix operations, taking its answers from Unix "man" manual pages. An information retrieval approach would use the words from the user's query to search the database of man pages and return those pages which are deemed to be relevant. This isn't specific enough for a help system, it leaves the user with a number of man pages to read without knowing where within each man page lie the relevant bits, if any. Additionally some of the retrieved documents will have

the right words scattered about, but still will not be relevant. A question-answering approach would have a comprehensive knowledge base of Unix operations over which it could inference, so as to generate the answer to the question. So far, this is practical for only very small domains. The ExtrAns system tries to have broader coverage than might be practical in a question-answering system.

Very briefly, the approach is to parse the man pages and build a database of logical forms. Associated with each logical form is a pointer to the segment of text it was derived from, so text needn't be generated from the database. The user's query is similarly parsed, and an inferencing mechanism compares it to the database, retrieving some of the stored logical forms. The result is that the segments of the text which produced the best matches are displayed to the user.

The ExtrAns system would seem to be a competitor to the Latent Semantic Analysis information retrieval technique used by AutoTutor, as described in Chapter 7. In AutoTutor, it is necessary to match a student's utterance to stored databases of sentences in order to determine their truth, relevancy, etc. It seems to me that the ExtrAns approach, if used in a system like AutoTutor, could be more precise in its evaluations. When judging whether a student sentence was relevant to the topic at hand, it might more exactly pinpoint what parts of the topic the student addressed and what remains to be tutored. On the other hand, a compelling advantage of Latent Semantic Analysis is that no parses and logical forms are generated from either the student utterances or the sentences (whole textbooks of them) which go into the databases.

Direct Memory Access Parsing

An approach to semantic parsing which captured my interest for a while is "Direct Memory Access Parsing," described by Charles Martin [1990], who calls his parser DMAP. It is a case-based parsing program which uses small amounts of syntax. A smaller version called Micro-DMAP is described by Riesbeck and Schank [1989].

All the case-based reasoning programs described by Riesbeck and Shank use a common content-addressed memory organization, which is composed of packets they call MOPs. MOPs are linked in several ways, first forming a kind of class-and-properties inheritance hierarchy (called the abstraction hierarchy) where each MOP inherits properties from its superclasses, second forming a kind of network of slot-and-filler records (called the packaging hierarchy), where each MOP can have a record whose slots are filled with other MOPs. MOPs are divided into "abstraction MOPs" and "instance MOPs," where one abstraction MOP can have several instance MOPs underneath it, instance MOPs cannot be abstractions of anything else.

In practice, MOP memory contains a kind of semantic network of abstraction MOPs, and information is added to this network by creating instance MOPs and filling in records. There is an associative search mechanism which, given a new MOP to add, can search the hierarchy to figure out where it fits. A power of MOP memory comes from the fact that the abstraction hierarchy can be changed on the fly. For example, if new data shows up that cannot fit into the existing semantic network (the abstraction hierarchy), the system can see how this new data differs from the existing data, create a new abstraction, and reorganize the semantic network.

There were two qualities which attracted me to DMAP. The first is that its parser is predictive and goal oriented, skipping constituents which are uninteresting, rather similar to the robust parser in SOPHIE. Second is that DMAP doesn't produce a "result" in the conventional sense, rather what it does is use incoming phrases to update the state of its MOP memory. As more words and sentences are parsed, the state of the memory is continuously updated.

This second feature is interesting in the abstract. Imagine the following bit of made-up dialogue:

S: CO increases. [We imagine this is wrong]
T: What are the determinants of CO?
S: HR.
T: And . . . ?
S: SV.
T: Very good.
But you predicted that SV decreases and HR doesn't change.
Do you still think CO increases?
S: Decrease.

In a MOP-based system, MOP memory contains a lot of state information structurally encoded. In particular, in CIRCSIM-Tutor it would contain the fact that CO is the variable which the student got wrong and is currently being tutored. Pre-coded structural information would show that CO is the kind of thing which has determinants, determinants being another kind of thing, and an instance of "determinants" would exist for CO, showing HR and SV. The tutor's question about "what are the determinants of CO" would set up a parallel set of MOPs with the same information, which the student's answer would have to match.

The student's answer is processed a piece at a time. In the above example, which is similar to behavior sometimes observed in our tutoring transcripts, the student's

answer can be split over several turns. In CIRCSIM-Tutor as it currently exists, a partial answer causes a difficulty. A subplan must be invoked to fetch the other half of the answer. In a DMAP system no such work is necessary. The student's words arrive in a continuous stream, updating the memory as they arrive. If the answer arrives in two separate turns, there should be no problem.

This small example is an illustration of the general observation that an intelligent tutoring system contains a lot of state information, and every student utterance changes its state.

Ultimately, after experimenting with DMAP grammars for simple questions and answers, it proved not very useful for CIRCSIM-Tutor. The big issue is that there is no MOP-based planner in the traditional sense, and traditional planning is how we think about the tutoring problem. This is not to say that case-based reasoning couldn't be used, I do not know that it can't. But DMAP is driven by the state of MOP memory and each input changes the state of MOP memory. In order to utilize it fully we would have to restart the CIRCSIM-Tutor project from the beginning with a completely different planning paradigm.

Another fundamental difficulty is that in a dialogue, each turn is an event. Though it be attractive to think of each utterance updating the state of MOP memory, in fact each utterance needs to be evaluated and responded to. The partial answers which arrive over several turns, as illustrated above, are a special situation.

CHAPTER VII

USING LATENT SEMANTIC ANALYSIS IN THE INPUT UNDERSTANDING TASK

A fairly new language processing technology, developed for information retrieval [Deerwester et al. 1990, Berry et al. 1994] but finding uses elsewhere, is Latent Semantic Analysis or LSA. A good introductory reference is [Landauer et al. 1998]. Using a technique similar to factor analysis, LSA builds a model to explain the occurrences of words in documents as linear combinations of factors.

Simplified Introduction to LSA

Suppose we have a collection of local news articles from the Chicago newspapers. Each article is taken only as a set of words, ignoring grammar, sentences, etc. Now we have some factors, say "education," "law enforcement," "politics," and "community organization." To every word we can assign a four-element vector, representing the strength of each factor in that word. For example, "chief" would have a high education component and a low law-enforcement component, while "commissioner" might have a low education component and a high law-enforcement component. The point is that the head of the schools is usually called the "schools chief" in the Chicago papers and the head of the police department is usually the "police commissioner." The word "chief" would have "politics" and "community organization" components derived from how often that and similar words occur in articles containing those concepts. A word like "petition" might have a large community-organization component, middling politics and

law-enforcement components, and a low education component, derived simply from the frequency with which it occurs in articles related to those topics.

Each word can thus be represented by a four-tuple, representing a point in factor-space. By summing (for example) the four-tuples of each word in a document, it is possible to assign to the whole document a four-tuple point in factor-space. An article containing many instances of both the words “chief” and “commissioner” could be about law enforcement and education both, and so on.

It is helpful to think of this document as a vector in factor space, starting from the origin. It may be a fairly long vector (because of many words added together) or a short one, but its direction will be determined solely by the relative contributions of the four components.

When using Latent Semantic Analysis for information retrieval the collection of documents is analyzed in this manner, producing factor-space vectors for each word and for each document. When a query is processed the query itself is treated as a document: the components of each word are looked up and combined to form a vector for the query-document. Then the vector representation of the query-document is compared to the vectors for all the stored documents in order to retrieve ones which are similar.

A common measure of similarity is the cosine of the angle between the vectors representing two documents. If your query contains only a few words, its vector may be quite short. Yet there should be a way to measure its similarity to documents, which (because they are wordier) may have longer vectors. If the factors in the query and a stored document are in the same proportion (e.g., equal parts education and law

enforcement), that document is likely to be one you want to retrieve. In this case the vectors point in the same direction, and the angle between the query vector and stored document vector is very small.

LSA in More Detail

The particular mathematical model used by Latent Semantic Analysis is called singular value decomposition. The thing being modeled is a matrix of word frequencies in documents: each row of the matrix represents a different word and each column of the matrix represents a different document. A cell of the matrix contains the count of the number of times one word occurs in one document.

In general, these matrices are rather sparse. The set of fifty K-series CIRCSIM-Tutor project transcripts (without spelling correction) contains roughly 3800 different words. There are 10,600 documents, where each sentence is one document. However each document contains only a few words, so the matrix contains 51,000 non-zero entries, for an occupancy of 1.3×10^{-3}.

If A is an $m \times n$ matrix (where m is the number of words and n is the number of documents), singular value decomposition produces three matrices U, S, and V, where:

$$A = USV^T$$

and U is an $m \times n$ matrix, V is an $n \times n$ matrix, and $U^TU = V^TV = I_n$ identity matrix. S is a diagonal matrix of values $s_1 \ldots s_n$ called the *singular values*.

Intuitively, the singular values are scaling constants, one for each column of U and corresponding column of V. The columns with the largest scaling constants contribute the most to the reconstruction of A.

If one picks only the k largest singular values, and the corresponding k columns of U and V, then one obtains a k-dimensional approximate model using the most explanatory dimensions. The reduced U is an $m \times k$ matrix called the *term vectors*, where one row of U represents the k components for one word. The reduced V is an $n \times k$ matrix called the *document vectors*, where one row of V contains the k components for the one document.

If we want to convert a query into a document vector, first form an m-element vector q of the counts for each word in the query. If we are using the CIRCSIM-Tutor project K transcripts q contains 3800 elements, one for every possible word. Since a typical query contains only a few different words, most of the elements are zero. The corresponding encoded document vector q' is given by:

$$q' = q^T U S^{-1}$$

The process is straightforward:

- Find the k-dimensional term vector for each word in the query
- Scale it by the number of occurrences of that word in the query
- Vector-add all the results
- Scale each dimension by the inverse of that dimension's singular value

The cosine between q' and any other document vector v is computed in the conventional manner:

$$\cos(q', v) = \frac{q' \cdot v}{|q'||v|}$$

Latent Semantics

In contrast to the simplified example I gave above, which has semantic components such as "education" and "law enforcement," the actual SVD technique induces components without regard to any pre-conceived semantic meaning. It may be possible to manually observe that all the education-related words or articles have high values for one particular component, and thus assign a semantic interpretation of "education-ness" to that dimension, but the model remains ignorant of such classifications.

In fact, words and documents with related meanings will tend to have similar vectors. A comparatively small number of dimensions effectively segregates documents by differences in meaning. But no lexicon or set of semantic primitives was ever used in the derivation of those components; the only data is frequencies of words occurring in documents. For this reason the semantic meanings of the words and documents are said to be "latent" in the original set of documents and discovered by the SVD procedure.

Sometimes the consequences of using latent semantics can be startling. It is possible to retrieve highly relevant documents which have *no* words in common with the query vector. If two different words tend to occur in similar contexts (taken across all the documents), then a query using one word might retrieve a document using the other. On the other hand, the fact that syntax and word order are ignored means that two queries "A increases B" and "B increases A" are considered identical.

The model produced by decomposing a set of documents (the term vectors, document vectors, and singular values) is called a *semantic space*. It makes little sense to

compare words and documents between semantic spaces. The meaning of a word in a semantic space is its vector components, but the components themselves have no meaning except that they are the ones which best fit this collection of documents and words.

Not much is known about picking the optimum number of dimensions k. Most important is that $k < n$, the number of documents in the semantic space. Without some reduction in the number of dimensions the original term-document matrix is perfectly modeled; the "meaning" of a word becomes an explanation of exactly how many times it occurs in each individual document. To see why this is undesirable, consider two very similar words "physician" and "doctor." A good semantic representation would predict that both words were almost equally likely to occur in some document which contained that concept. More generally, it would predict an ideal term-document matrix where (almost) synonymous terms were (almost) equally likely in any given context. However suppose document number 17 was written using "doctor" several times, but no "physician," while semantically similar document number 18 made the opposite choice. A model which exactly reconstructs the observed term-document matrix contains distinctions which explain the observed occurrences of "doctor" and non-occurrence of "physician" in document 17, and the opposite behavior in document 18. In fact, for information retrieval purposes, we would like the two words to be equivalent. Any model which explains differences between the two will have lower recall. Between any two synonyms this problem will occur: the more the model attempts to explain why one word was chosen (perhaps randomly) over another, the less useful it may become.

Generalizing, the reason not to exactly model a particular term-document matrix is that the set of documents being modeled can be thought of as a statistical sample drawn from an ideal set, thus differing from the ideal. Explaining these differences is counter-productive. Furthermore, consider that there may be biases in usage which you actively desire not to model. “Physician” may happen to be used preferentially in some population of documents, thereby biasing the model, but if “doctor” occurs with any frequency you may very well want your information retrieval query to ignore the bias.

If the number of dimensions in the model is equal to the number of documents, you could assign one dimension to each document and exactly reconstruct the term-document matrix. Singular value decomposition produces the same effect, with the difference that it constructs the dimensions to be orthogonal in a way which can be ordered from most explanatory to least explanatory. This means that the most explanatory dimensions are, as a practical matter, capturing the semantic distinctions which most explain the term-document frequency matrix while the least explanatory dimensions are, in effect, forcing particular words into particular documents. Thus picking the $k < n$ most explanatory dimensions can produce a more useful model.

There seems to be no principle for picking k. In their early paper, Deerwester et al. [1990] reported good results with $k = 100$ in two information retrieval experiments: one on a 5823 term, 1033 document set of medical abstracts and one on a 5135 term, 1460 document set of computer science abstracts. More recently, Landauer and Dumais [1997] performed word synonym experiments using a semantic space of articles (truncated to 2000 characters) from an encyclopedia intended for young students, for a total of 30,473

documents. Testing increasing sizes of model up to 1032 dimensions, plus the full unreduced model, they discovered a broad peak around $k = 300$.

LSA and Psychology

One view of information retrieval is that it is a kind of stimulus-response process of the human organism, where the stimulus is the occurrences of terms in documents and the response is a judgment about a document's meaning or the relevance of a document to a query. This thinking informs various term-weighting schemes used by a number of information retrieval algorithms. Dumais [1991] experimented with applying several such schemes to LSA,[3] showing that it can be made considerably more effective in this way. For Latent Semantic Analysis, term-weighting starts by scaling each count in the term-document matrix. $A_{i,j}$, the number of occurrences of term i in document j, is replaced by $\log(A_{i,j}+1) \;/\; \mathrm{G}(i)$, where $\mathrm{G}(i)$ is the entropy of term i across all documents. Psychologically, this is modeling an organism's logarithmic response to the strength of a stimulus and its direct response to the specificity of the stimulus. The stimulus in this case is the frequency of occurrence of a term in a document. Specificity is modeled by inverse entropy; high entropy for a word means that it occurs equally frequently in all documents, which makes it not very specific. Recent papers on LSA all make use of these transformations, although I have not yet incorporated them into my own experiments.

There has been considerable interest in using Latent Semantic Analysis as a model for human acquisition of word meanings. Landauer and Dumais [1997] explore *Plato's*

[3] Dumais [1991] makes no reference to the possible psychological validity of term-weighting, saving such discusion for [Landauer and Dumais 1997].

Problem, the question of how people learn as much as they do given an apparent insufficiency of stimulus. In particular they were interested in the acquisition of vocabulary. Anderson and Nagy [1993] estimate that the average high school senior has a reading vocabulary of about 40,000 different words, having learned 2,000 to 3,000 new words per year throughout much of elementary and high school. In their accounting, words whose meaning could be guessed from morphology or compounding were not distinct, thus "cleverness" is not a new word to a student who knows "clever," but "busy" is distinct from "business" because knowing one does not confer a knowledge of the other. The bases for counting words as distinct are described in their earlier work [Nagy and Anderson 1984]. Anderson and Nagy admit that their estimate is low, omitting the proper nouns which constitute a large fraction of ordinary vocabulary (they cite "Methodist," "Amazon," "Republican," "Egypt," and "Platonic" as examples) and not counting the variety induced by polysemy. Landauer and Dumais also note that 40,000 is at the low end of published estimates. They cite studies that show that 60% of words found in newspapers, primarily proper nouns, do not occur in dictionaries, and are therefore (due to experimental design) excluded from most measurements of students' vocabulary. In any case there is no evidence that students are directly instructed in the meanings of so many words; typically an elementary school class may receive definition-based instruction of 300 words per year of which a student learns 200 [Anderson and Nagy, p. 12]. Neither do students look up many thousands of words in dictionaries every year.

In short, it seems likely that people acquire their vocabularies largely from context, from reading and listening. Recall that Latent Semantic Analysis builds its models entirely from text; it never uses a dictionary of terms, relationships, or any other external source of semantic knowledge. Can the behavior of LSA possibly model human behavior?

One of the questions asked by Landauer and Dumais was the following: do LSA models learn vocabulary at the same rate that humans do? "Learning" in this case means how much vocabulary a model "knows" versus how much text went into the construction of the model. For this investigation they obtained a set of eighty word-matching questions used by the Test of English as a Foreign Language. A question consists of a target word and several candidate words, the student must pick which of the candidates is closest in meaning to the target word. This task is readily suited to LSA. The measure of a model's vocabulary was taken to be the score on the exam. The semantic space was generated from text from a student encyclopedia, the amounts of text used in model construction were chosen to represent the amounts of text that a child might have cumulatively read by various ages. Landauer and Dumais conclude that LSA models can acquire vocabulary at the same rate that children do, and in the end it performs as well on the TOEFL vocabulary exam as adult test-takers. Furthermore the pattern of mistakes mimics the human pattern of mistakes; when the model picked incorrect candidates, the ones it picked correlated well with the mistakes that humans made.

From the perspective of a computer scientist these results are very encouraging. They obviate the need for a variety of specialized and sometimes baroque constructs postulated by linguists and cognitive scientists in order to explain innate language ability.

These mental structures are tailor-made for correctly inducing grammatical distinctions, semantic distinctions, and so on. Pinker [1994, pp. 149 et seq.] has a typical discussion of the vocabulary problem, estimating that an average person learns a new word every ninety waking minutes from the ages of one until eighteen. Again, that excludes words whose meanings can be derived from other words. We would not expect the average person to be able to learn and retain that many phone numbers, for example, especially without explicit memorization. However there is evidence that when presented with new unexplained or wholly invented words, children are frequently able to ascribe useful meanings after a surprisingly small number of encounters. Pinker describes a thought experiment which illustrates the problem:

> There is one more reason we should stand in awe of the simple act of learning a word. The logician W. V. O. Quine asks us to imagine a linguist studying a newly discovered tribe. A rabbit scurries by, and a native shouts "Gavagai!" What does *gavagai* mean? Logically speaking, it needn't be "rabbit." It could refer to that particular rabbit (Flopsy, for example). It could mean any furry thing, any mammal.... It could mean scurrying rabbit, scurrying thing ... or scurrying in general... or "It rabbiteth," analogous to "It raineth."
>
> The problem is the same when the child is the linguist and the parents are the natives. Some how a baby must intuit the correct meaning of a word and avoid the mind-boggling number of logically impeccable alternatives.
>
> [Pinker 1994, p. 153. Many alternatives elided for brevity.]

Pinker concludes that there must be specialized word-learning mechanisms in the mind which are pre-disposed toward some varieties of distinguishing characteristics, concluding "we all get away with induction because we are not open-minded logicians but happily blinkered humans, innately constrained to make only certain kinds of guesses—the

probably correct kinds—about how the world and it occupants work." Later he mentions various experiments that tried to tease out what some of those inductive biases might be.

The clear conclusion of the LSA vocabulary-learning studies is that it is possible to learn words at human rates without an elaborate specific word categorizing mechanism. These experiments do not show what human learning looks like—there is no reason to believe that people are performing singular value decomposition in their heads—what the experiments show is that there is enough contextual information in ordinary text to make vocabulary learning possible.

Tied up with the vocabulary learning problem is an epistemological question: what does it mean to "know" a term? For Landauer and Dumais, their model knew a word if it successfully picked the word nearest in meaning from a set of alternatives. By that criterion knowing a word means being able to identify its relationship to other words, as opposed to being able to write a definition or use it in a sentence. Measurements of human reading vocabulary are usually obtained by similar means: match a word with its antonym, place a word in one of several categories, identify which word is represented by *x* in "A is to B as C is to *x*." Nagy [1995] notes that repeated exposure to words in context deepens a person's ability to distinguish between polysemous senses, know the register in which a word may be used, make finer distinctions with other words, etc. A lot of vocabulary knowledge seems to pertain to the relationships between words. LSA models well the human behavior of deepening knowledge of lexical distinctions with increased exposure. On the other hand Nagy also argues that vocabulary learning needs some grounding. As he puts it "there may be theoretical reasons to distinguish linguistic

knowledge from world knowledge; but I would not recommend that a teacher avoid mention of trunks in a discussion of the word 'elephant' on the grounds that having a trunk is not a logically necessary property of elephants." LSA doesn't model this very well; were it not induced from the term-frequency matrix, an LSA model wouldn't "know" there is a relationship between elephants and trunks.

To give more concreteness to the epistemology question, consider the words "soffit" and "fascia." I know that they are architectural terms, name components which occur near each other, and are frequently related to the underside of the eaves of a house. As measured by a general-knowledge pick-the-nearest-term vocabulary test (such as might occur on the TOEFL) I probably know these words. On an architectural vocabulary exam I would surely flunk. An LSA semantic space built from general texts would probably model my own knowledge of these two words, while a semantic space built from the appropriate architecture textbooks would model the expert's knowledge.

From the standpoint of writing an intelligent tutor, the experiments that show LSA learns vocabulary as well as humans do (as measured by tests) are extremely useful. It means that using an LSA-generated semantic space for understanding human input, without a conventional lexicon of word meanings, may be a plausible approach.

Application of LSA in AutoTutor

There is one intelligent tutoring system which incorporates Latent Semantic Analysis into its input understanding component, namely AutoTutor [Graesser et al. 1998, Anwar et al. 1998, Wiemer-Hastings et al. 1998] from the Tutoring Research Group at the University of Memphis. AutoTutor teaches basic computer literacy. One of its

communication modes, in addition to animated diagrams and vocalization, is written dialogue. A primary object of AutoTutor is to exploit "subjective construction of explanations" on the part of the student. To put this more concretely, AutoTutor asks open-class questions such as "What is the function of the CPU?" and "What are the differences between RAM and ROM?" The student formulates a free-text reply possibly several sentences long.

Partly from observing untrained college tutors, the Tutoring Research Group makes the following observation: even without using sophisticated strategies, tutoring via dialogue can still be extremely effective. We would like to promote hinting and Socratic dialogue in CIRCSIM-Tutor, dialogue strategies which have proven to take considerable effort to implement. The AutoTutor project is testing the hypothesis that if they ignore the sophisticated tutorial strategies and engage their students in simpler dialogue, useful learning might result.

In AutoTutor the computer literacy curriculum is divided into three topics: hardware, operating systems, and the Internet. Each of these topics is divided into 12 subtopics to teach. Vastly simplifying (there are extensive curriculum scripts, curriculum planning production rules to pick subtopics, and animated graphics scripts), AutoTutor teaches a subtopic by presenting a question or problem and accepting the student's response. Then the response is evaluated as to truth, relevance, and proximity to various stored right and wrong responses and a set of possible student questions. Depending on these calculations, and after consulting a script, AutoTutor can issue an acknowledgment

(positive, negative, or neutral), possibly hint, possibly prompt for more information, issue a summary, and so on. Teaching one subtopic typically requires several turns.

Since questions such as "What is the function of the CPU" can admit of a wide variety of answers, many of which will be misguided, it is clear that producing a symbolic semantic representation of the free-text, multi-sentence utterance could be a tall order. After several stages of processing (including recognizing some short answers and frozen expressions, and recognizing dialogue acts via a neural network), AutoTutor usually doesn't attempt to parse or produce a symbolic representation of a student's lengthy input. Instead, it computes an LSA vector-space representation. The semantic space was derived from the relevant sections of two textbooks, several articles on computer literacy, and content specifications for all the topics and subtopics that AutoTutor teaches. Then AutoTutor performs the following tests:

- To determine whether the student's answer is true or not, the maximum of the cosines compared to every document in the semantic space is computed. Notice that a student's answer might be judged as true even if it is not relevant to the subtopic at hand.
- To determine whether the student's answer is relevant, it is compared to the all the documents relevant to the topic being taught.
- Comparing the student's answer to the vector representations of various known bad answers tells whether the student needs certain remedial information or certain hints.
- Similarly, the student's answer can be compared to various known good answers.

The results of all these comparisons are used for choosing AutoTutor's next dialogue move.

It seems to me that the primary reason the LSA approach works for AutoTutor input understanding is that computer literacy is a vocabulary course; the students are expected to learn vocabulary and concepts. For this curriculum the student shows knowledge of the topic by being able to define concepts or state which concepts are relevant to a particular situation. Unless the student is acting perversely, the words and phrases uttered ought to be a pretty good indicator of the concepts and relationships the student is trying to convey. It should also help that the student has been exposed to the subject; tutoring happens after the student has read the book or attended the class, so the student's language is more likely to contain terms from AutoTutor's LSA semantic space.

The question of vocabulary epistemology is important here. One might suppose that properly responding to a student's explanation of "What are the differences between RAM and ROM?" demands detailed semantic descriptions of words such as "RAM," "ROM," "storage," "read," and "write." Said descriptions could then be used to construct a semantic representation of the student's utterance, which could be reasoned about or compared with stored semantic representations in order to make judgments. What an LSA model knows about "RAM," "ROM", "memory," "read," and "write" is the differences in lexical contexts in which they occur (or the latent semantics derived therefrom). For AutoTutor's purpose this is just enough and not too much. If the semantic space is constructed from documents relevant to the topic then the LSA representation of the

student's utterance can be compared to various classifications of known utterances, thereby picking a classification for the student's answer with fair confidence.

In the demonstration that I witnessed it was notable that AutoTutor ended the dialogue for each subtopic with a summary of the material to be taught. If this is AutoTutor's general behavior (I have not seen it documented), it seems to be a safe strategy. In the event that AutoTutor does not correctly classify and respond to a student's answer the final summary ensures that the student is left with a correct explanation.

Diagnostic Questions: A Potential Application of LSA in CIRCSIM-Tutor

For the normal answers to the questions that CIRCSIM-Tutor version 2 currently asks there seems to be no advantage to applying Latent Semantic Analysis. Partly this is because the questions are so closed. All the questions are readily answered with a few words at most. Even when the student gives a longer utterance in response to a question, the answer can almost always be recognized from a few words within it. Furthermore there are syntactic issues. For example the "is" confusion—copula vs. inotropic state—is a syntactic issue, and thus would have to be solved by other means. It is easy to find examples in the human tutoring dialogue where syntax is important within exchanges which are no more complicated than ones CST v. 2 currently participates in. Here is an example of a potential problem:

T: What does CC affect?
S: SV
* T: Sure. So what happens to SV?
* S: If CC d, then SV d.
T: You got it. [K5-89]

The starred response cannot be understood if it is treated as an unordered bag of words as LSA would treat it. To see this, imagine the student had typed "If CC i then SV d." This utterance would contain two parameters and two opposite changes (one increase and one decrease), but there would be no way to distinguish which increased and which decreased. It might be possible to notice the causality from the occurrence of the words "if" and "then" but have no way to infer which is causing which. Unlike computer literacy, the subject of AutoTutor, the baroreceptor reflex is not a vocabulary course: the distinctions which are syntactically marked can be the heart of the matter.

Nevertheless I think that CIRCSIM-Tutor v. 3 presents opportunities for taking the LSA approach. The key observation is what Hume [1995, p. 73] called the "information gathering question" (I will call it a diagnostic question) which the tutor asks right after the predictions are collected, eliciting an explanation from the student Here are two examples:

T: OK. That completes your DR predictions. Most of them are correct. However I want to pursue IS with you. Can you tell me what you think IS means? [K47-tu-56]

T: Let's talk about your answers here. Why do you think that tpr will increase? [K45-tu-54]

Noting that the tutors often have difficulty explaining their own intentions, Hume provides three possible explanations as to why these questions are asked:

- Provide the student an opportunity to re-think and correct an answer
- Provide the tutor useful information about the student's misconceptions
- Provide some kind of hint

Perusing examples in the transcripts, it seems that the student's response is often ignored, which makes it hard to divine what the tutor's intention was.

Let me propose two more reasons why asking these diagnostic questions might be useful:

- Stimulate the self-explanation effect
- Provide a polite context before possibly informing the student of a mistake

I have not found any attestation for this last reason in the tutoring literature. Nevertheless I make the claim, substantiated only by introspection, that there can be powerful psychological and social reasons why asking the diagnostic question is polite. Firstly, as anybody with teaching experience knows, students often want to explain themselves after they have made a mistake. Indeed, sometimes it can be hard to stop them. Even though the diagnostic question might have other uses, asking it can recognize and assuage the student's desire to explain. Secondly, in normal conversation it is often polite and more effective to ask people for their opinion before telling them your own. They are more likely to hear you if you ask them their own opinion first. In a similar vein, it is possible that asking the question makes the person more invested in knowing the answer. I claim that when tutors ask these diagnostic questions at the start of teaching a new topic it is an analogous behavior.

Notice that of the five possible reasons for asking diagnostic questions, four of them do not require the tutor to have a good understanding of the answer. For providing the student an opportunity to re-think and correct, for providing a hint, for providing an opportunity to self-explain, and for providing a polite context, the value of the question lies in the side effects it has on the student. Only for acquiring information about the student's misconceptions is a good understanding of the answer needed.

Graesser [1993] provides evidence that tutors rarely use the results of diagnostic questions. One part of that study was comprised of 44 one-hour sessions of face-to-face tutoring of undergraduates in a psychology research-methods course, taught by three students who were successful in the equivalent graduate-level course. The college tutors devoted 14% of their questions to "get[ting] student to justify something, explain something, or generate an example," but only 2% of the tutor turns after a question was answered were categorized as "tutor diagnosis, dissects, or remediates student errors." Nevertheless, despite the paucity of deep diagnoses of student misconceptions, the tutors were shown (by pre- and post-tests) to be quite effective.

Given that asking diagnostic questions might be beneficial, even absent precise understanding of the student responses, I propose that we add diagnostic questions to CIRCSIM-Tutor. The idea is to use an LSA mechanism similar to AutoTutor's to process the results.

First we catalog every diagnostic question in the transcripts, noting the question, the answer, the tutoring context, and how the tutor responded. From these we create a set of scenarios, containing the question, the points in the tutoring when that question gets asked, the categories of answer, and the actions prompted by each category. These scenarios are small versions of the AutoTutor scripts. The final step is to build a bundle of documents for each category of answer; LSA will be used to compare the student's utterance with these documents in order to categorize it. Once categorized, a suitable response can be generated.

Crucial to this technique is the observation that frequently in our transcripts the content of the student's answer to such a question is ignored, especially if the answer was long and involved or if it was dramatically wrong. In the proposed implementation, common easily expressed conceptual mistakes will be cataloged, they will each have a separate category and response. Correct answers likewise can be responded to, although the tutor should echo back a summary just in case there was a mistake in classifying the answer as correct. Involved or dramatically wrong answers will fit in a category which is essentially ignored. Thus it may be possible to effectively mimic human tutoring responses to these diagnostic questions by using only the limited "understanding" of student responses that LSA can provide.

One advantage we have in incorporating this experiment into CIRCSIM-Tutor is that the diagnostic questions are optional. They would be grafted into the normal tutorial dialogue at the point that human tutors would ask them. But if some of the scenarios are hard to define, or are impractical, or result in drastic revisions of the tutorial plan, we can simply not include them in the tutorial plans at that point, and the CST planner will behave as it currently does.

Experiment using LSA for CIRCSIM-Tutor

I've been experimenting with using the K-series transcripts as a semantic space, seeing what results from various queries, trying different numbers of dimensions, etc. One such semantic space has the whole unedited set of K-series dialogue. Another has just the student turns, and another has spell-corrected student turns.

As an illustration of matching a hypothetical student utterance to a CIRCSIM-Tutor semantic space, Table 5 shows the sentences in the CIRCSIM-Tutor transcripts that are closest to the query "Frank Starling."

In this experiment, each "document" was a single tagged sentence from the first fifty K-series transcripts. The resulting matrix had 3823 words and 10251 documents (sentences). The transcripts were not spell-corrected, which I estimate meant that there were about 900 nonsense words, reducing the accuracy. Neither were the words

Table 5. Sentences Closest to "Frank Starling"

Rank	Cosine	Tag	Text
1	0.751	K5-tu-21-2	Very little filling takes place in thisperiod.
2	0.742	K9-st-36-2	Isn't the amount of filling equivalent to the preload?
3	0.729	K9-tu-35-3	It's the effect of increased filling on contractility.
4	0.709	K26-tu-54-2	You are confusing the Frank-Starling effect (increased filling gives greater output) and contractility.
5	0.705	K41-tu-72-2	It's the Starling mechanism.
6	0.702	K35-tu-44-2	Sv is most importantly determined by filling of the ventricle (Fra nk-Starling effect) thus the mt potent determiner is rap.
7	0.691	K9-st-24-8	Thus, the question becomes -- what effect
8	0.680	K9-tu-35-7	The change in performance that is associated with a change in filling is Starling's effect, the length/tension effect.
9	0.672	K40-tu-54-4	We call that the Frank-Starling Effect
10	0.667	K3-st-46-1	Filling decreases
11	0.648	K8-st-32-1	Decreased filling time between contractions
12	0.630	K2-tu-26-2	Remember thatmost of the filling takes place early in diastole and very littlelate in diastole.
13	0.628	K37-tu-98-3	So you need to keep it distinct from the frank-starling effect.
14	0.616	K8-tu-33-1	This effect is really very small until you get to very high HRs

lemmatized, so “increases” and “increasing” were counted as separate words. I estimated the number of nonsense words by sampling approximately 200 of the almost 2000 words which occurred only once in the transcripts. In this experiment I used a 70 dimension model, and the psychologically-inspired term-weighting adjustments (described above) were not applied.

In this and the other experiments I used the SIS2 singular value decomposition routine from the SVDPACKC package [Berry et al. 1993].

For the “Frank Starling” query note that the three sentences judged closest contain neither of the query words, yet they are quite relevant. (The Frank-Starling effect dictates that when the heart’s chambers are filled with more blood, distending the muscle fibers more, the heart exhibits more contractile force.) Note also that there are considerable spelling errors among the retrieved sentences, such as “thisperiod” joined into one word and “Fra nk” sporting an interstitial space. Despite the dirtiness of the data, which lowers retrieval precision, the experimental queries almost always returned relevant documents.

Notice that sentences containing the word “effect” are strongly related to the query “Frank Starling” in this semantic space, causing a number of false hits. This comes from the prevalence of the phrase “Frank Starling effect.”

The experiments with using LSA did confirm that there are important semantic distinctions which the Latent Semantic approach is not capable of making. The two sentences “A increases B” and “B increases A” are equivalent, as expected, because

syntax is ignored. This may make it difficult for the LSA approach to understand answers which involve causal reasoning, for instance.

In a similar vein, note that two words which occur in the same contexts will be analyzed as being quite similar. In this semantic space “increase” and “decrease” are such a pair; in the transcripts any sentence containing the one word could just as well contain the other. Queries involving the one thus frequently retrieve sentences containing the other. It is not clear to me why this should be a limitation, however.

CHAPTER VIII

CIRCSIM-TUTOR V2.5 ENHANCED INPUT UNDERSTANDER

This chapter contains a description of the new input understander which is currently running in CIRCSIM-Tutor version 2. Since many aspects of the system have been significantly upgraded in the past year we call it version 2.5. It is my intention that the same input understanding code will be able to run in the new CST v. 3.

In contrast to the previous CIRCSIM-Tutor input understander (see Chapter 2), the fundamental idea in the new one is to be as permissive as possible. The new input understander extracts whatever is needed from the student's input and ignores the rest. Only a little parsing is done; mostly the understander is searching for words and phrases which could be answers to the question that CST just asked. It is possible to "fool" the understander, viz.:

T: Which determinant is dominant in this case?
S: Anything except HR.
T: HR is correct.

As a practical matter this is not a problem. In the log files of student instruction since the new understander was introduced, I found no instances of this permissiveness causing a student's input to be understood incorrectly.

Context of the Input Understander in CIRCSIM-Tutor v. 2

In CIRCSIM-Tutor version 2, the input understander is called directly after a question is asked. It is given a logic form which is usually the same logic form as was given to the text generator to emit the preceding question. The understander returns a logic form (usually) or a simple list (in a few instances) containing the student's answer. The

logic form is the sole input to the understander, which runs as a subroutine without side effects. In particular, the understander has no knowledge about the state of the tutorial dialogue beyond what was the immediate question, and leaves no result beyond the answer it returns. Figure 4 shows the questions asked to the student, the logic forms used for generating each question, the logic form passed to the input understander (if different), and the form of the answer returned by the input understander. Some of these questions are expressed in different words when they are re-asked after a student's mistake, but the content doesn't change.

It is important to realize that the capabilities of the input understander are limited by the rest of the CIRCSIM-Tutor v. 2 software. There are only eleven short-answer questions that CST v. 2 can ask the student. Due to haphazard accretion of code changes, before the new input understander was added the answers to some of those questions were processed by *ad hoc* code bypassing the input understander, so the actual variety of questions for which the old input understander handled answers was even less. The task of categorizing student answers is handled by the student modeler, which until recently recognized only "correct" (exactly what was expected) and "incorrect" (anything else which made it past the input understander). The discourse planner had provisions for only those two categories, so having the input understander recognize nuances such as various near misses was not possible.

Which determinant is dominant in this case?
(QUESTION (ACTUAL-DETERMINANT <var>))
(ANSWER (ACTUAL-DETERMINANT <var>))

Which variables are changed by the reflex?
(QUESTION (AFFECT REFLEX VARIABLE)) [generator]
(QUESTION (REFLEX *IS*)) [understander]
(<varlist>)

Which of the variables in the prediction table are determinants of <var>?
(QUESTION (AFFECTED-BY <var>))
(ANSWER (AFFECTED-BY <var> (<varlist>))

Will the reflex compensate for the change in Mean Arterial Pressure in DR?
(QUESTION (COMPENSATE REFLEX CHANGE *MAP*))
(<y-or-n>)

Will the reflex overcompensate for the change in Mean Arterial Pressure in DR?
(QUESTION (OVERCOMPENSATE REFLEX CHANGE *MAP*))
(<y-or-n>)

By what mechanism is <var> controlled?
(QUESTION (MECHANISM *IS*))
(ANSWER (MECHANISM (<mech>) <var>))

Is the relationship from <var-1> to <var-2> direct or is it inverse?
(QUESTION (RELATION <var-1> <var-2>))
(ANSWER ((<rel>) <var-1> <var-2>))

What stage must the value of <var> follow in SS?
(QUESTION (FOLLOW <var>))
(<stage>)

What variable is regulated by the baroreceptor reflex?
(QUESTION (REGULATE BARORECEPTOR-REFLEX VARIABLE)) [generator]
(QUESTION (BARORECEPTOR-REFLEX *MAP*)) [understander]
(<var>)

What is the correct value of <var>?
(QUESTION (VALUE <var>))
(ANSWER (VALUE <var> <val>))

What is the value of <var> in DR?
(QUESTION (VALUE-DR <var>))
(ANSWER (VALUE-DR <var> <val>))
There are two more logic forms for RR and SS similar to VALUE-DR.

Figure 4. CST v. 2 Questions, Logic Form(s) and Input Understander Results

Legend: <var> is a parameter, <varlist> is a list of parameters, <val> is one of +, –, or 0, <mech> is a mechanism of control, e.g. NEURAL, PHYSICAL, or STARLING, <rel> is POSITIVE or NEGATIVE, <y-or-n> is Y or N,

The CST v. 2 student modeler and discourse planner have recently been enhanced to be able to handle more cases[4]. The same eleven questions are asked, but a wider range of answers is permitted. For example, the question "by what mechanism is <var> controlled?" formerly admitted only answers which could be construed as naming a neural mechanism. Now the planner admits answers which can be construed as referring to the Frank-Starling relationship (which is a near-miss for the control of Inotropic State), arteriolar radius (a near-miss for the control of Total Peripheral Resistance), and other non-neural mechanisms. The planner can emit a considerable number of new hints, which means that it has more alternatives when the student makes a simple mistake. These changes have made for a tremendous increase in the usability of CIRCSIM-Tutor, and made it possible to use the program on classes of medical students.

Concomitant with the enhancements to CIRCSIM-Tutor's planner and student modeler, and partly to match their new capabilities, I replaced the input understander.

Overview of Processing

Figure 5 is a block diagram, showing the sequence of processing and the important data sources in the new input understander. The stages of processing are as follows:

- Regularizing the input string, dividing it into (putative) words.
- Lexical lookup, producing a list of lexicon entries
- Spelling correction (combined with lexical lookup)

[4] Yujian Zhou deserves most of the credit for upgrading the pedagogical capabilities of CST v. 2.

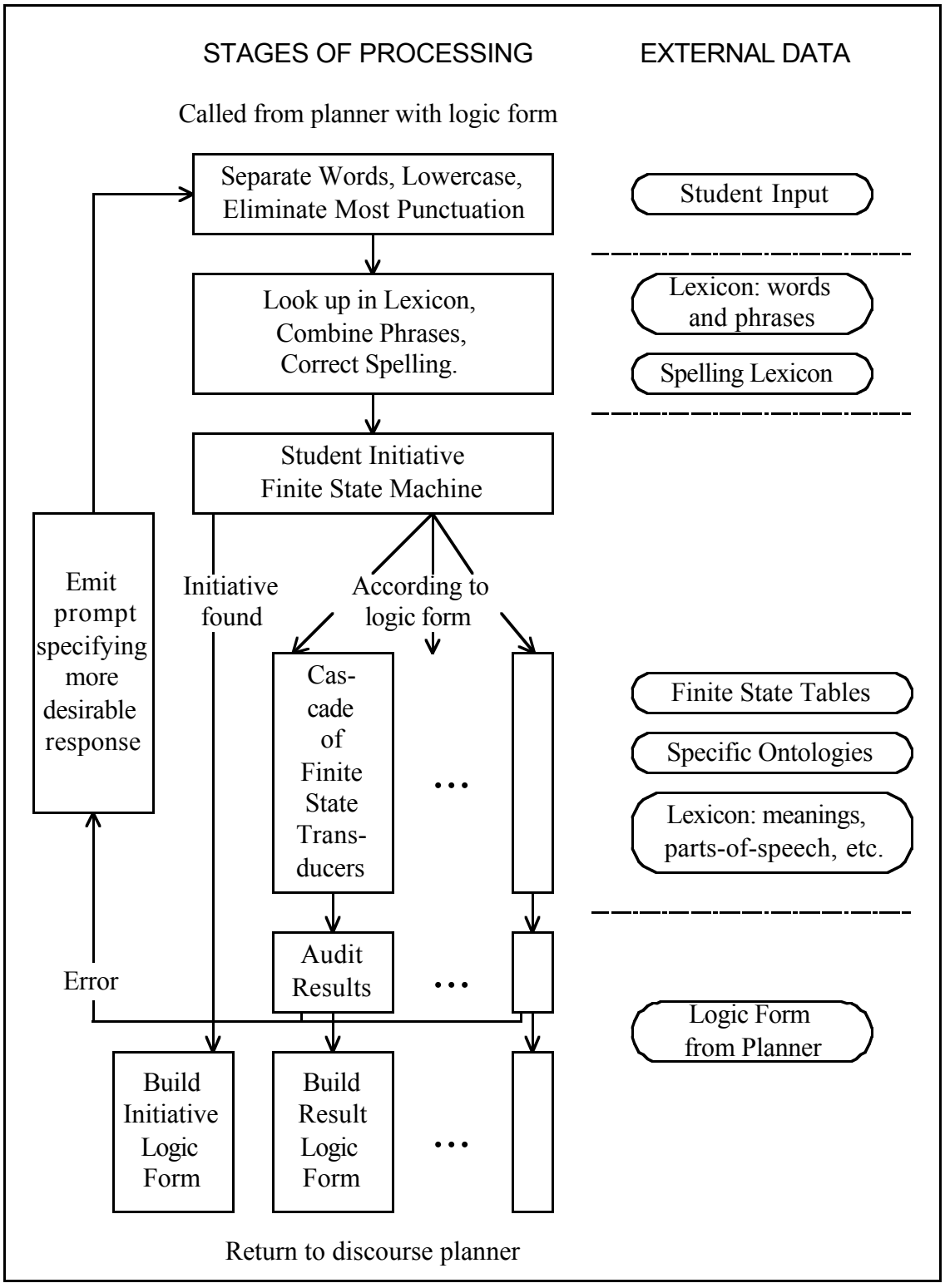

Figure 5. CST v. 2 Improved Input Understander Processing

- Processing for student initiatives and hedges with a finite state transducer
- Processing with a cascade of finite state transducers, using different sets of transducers for different questions which the student could be answering
- Checking the results of the transducers for errors and producing a logic form to return

The input understander in CST v. 2.5 is called as a subroutine and given the logic form for the question which was just asked. It returns a logic form with the answer.

Regularizing the input string consists of converting alphabetic characters to lower case, eliminating much of the punctuation, and dividing up into words. No lexicon is consulted, words are delimited by the "white space" in the input string.

Lexical lookup and spelling correction are described below.

The finite state transducers are also described below. This is a technology borrowed from information extraction (see Chapter 6) which can accomplish light parsing and extract from the text specific elements which are being looked for.

The stage of the input understander which checks for errors and produces logic forms is largely *ad hoc* code. The results produced by the finite state transducers are generally fairly convenient for use, so this is not as ugly as it could be. For example, if the question was to name a list of parameters, the transducers extract a list of parameters. Converting this list to the logic form which is to be returned to the planner is straightforward.

Error-checking is partially accomplished by examining the final state of the transducers. However the transducers could be more helpful with the error checking than they currently are. For example, if the acceptable answers are "directly proportional" or

"inversely proportional", the final state of the transducer will tell if at least one such answer was extracted. Unfortunately the transducer will not tell you if both answers were encountered in the same utterance, which is an error. Currently *ad hoc* code checks the output of the transducers for that and similar cases, but the transducers will be updated to check for errors like these in the future.

Finite State Transducer Mechanism

The central mechanism in the new input understander is a cascade of finite state transducers, with potentially a different set of transducers for each type of expected answer.

Cascaded finite state transducer technology is frequently used in information extraction systems, a good example is FASTUS [Hobbs et al. 1997]. Finite state machines are popular because they are fast and modular. They run in linear time with the length of the input; most algorithms based on context-free grammars are considerably slower. It is possible to combine finite state machines mechanically, producing machines for some combination of the languages processed by each machine. In the cascaded transducer model, each machine produces an output, which is usually some modification of the input. For example a transducer which looks for noun phrases might take as input a list of words (an utterance) with parts of speech marked and emit the same utterance with markers for the beginning and end of the NP. Then this result can serve as input to the next transducer, which might (for example) be looking for coreference between the phrases found by previous processing. Many aspects of the finite state approach to language processing are described in [Roche and Schabes 1997].

The transducers in the new input understander are basically ordinary non-deterministic Mealy machines (cf. [Hopcroft and Ullman 1979]), where every transition arc from one state to another contains a symbol to emit. The finite state machines can be described as follows:

- A state machine consists of a set of states plus a comparison function.
- A state consists of a state name, an ordered list of arcs, and a deterministic flag.
- An arc consists of a label which must be matched against a symbol from the input string, a string of output symbols to emit when the label is matched , and the name of a new state to enter when the label is matched.
- The comparison function is used for matching input symbols against arc labels.

The interpreter for these machines, a Lisp function, takes as input a state machine (a Lisp structure), an input string (a list of symbols), and the name of the initial state. It iterates over the input string, making state transitions and collecting the emitted symbols until the input string is empty. The machines being interpreted can be non-deterministic, meaning that more than one (or even none) of the arcs from a state might simultaneously match the current symbol in the input string. Because of this potential non-determinism the interpreter does not maintain a single current state. Rather, the interpreter mechanism maintains a set of all possible "routes," where each route contains a possible current state together with the list of symbols emitted en-route to that state. There are no states specially marked as "accepting"; interpretation ends when the input string is exhausted. The set of routes is the result.

The input string is a list whose “symbols” are any kind of Lisp objects. The comparison function is invoked to compare the label on an arc to an item in the input list without reference to the type or structure of a symbol. In the new input understander, one symbol in the input string is typically an entry from the lexicon containing a word, its meanings, parts of speech, etc. The symbols to emit on each state transition can also be any Lisp data object. Optionally there can be a function which is invoked to produce the output symbols.

Each state can be marked as deterministic. When a state is deterministic, only the first arc which matches the input symbol is taken, other possible transitions are ignored. In place of having a label, an arc can be marked so it is always transited when encountered.

An example state/transition diagram illustrating many of these points is Figure 6, the finite state transducer for copula deletion (below).

Some Finite State Transducers

Copula Deletion. This finite state machine is responsible for eliding the finite form of “to be” from student utterances such as “SV is not changed” and “it is SV.” The output of this machine is the same sentence as the input with copula “is” elided. The point of this exercise is to distinguish between copula “is” and physiological parameter “is,” the latter being the common abbreviation for Inotropic State. This machine will not alter sentences such as “is increased,” but “sv is increased” will be reduced to “sv increased.” Thus, of course, “is is increased” will be reduced to “is increased.”

The basic pattern of this machine is to look for the sequences:

{noun} + “is” + {optional negation} + {past/present participle or adj or adv}
{pronoun} + “is” + {anything}

This transducer is the first one in many of the cascades. It is a good illustration of the style and power of the cascaded transducer approach: it uses limited syntax to accomplish a limited but useful transformation of the text. After its operation, succeeding stages can assume that any occurrence of “is” refers to the physiological parameter.

Parameter and Qualitative Change Extraction. There are several machines here which basically perform keyword extraction, looking for parameter name, for

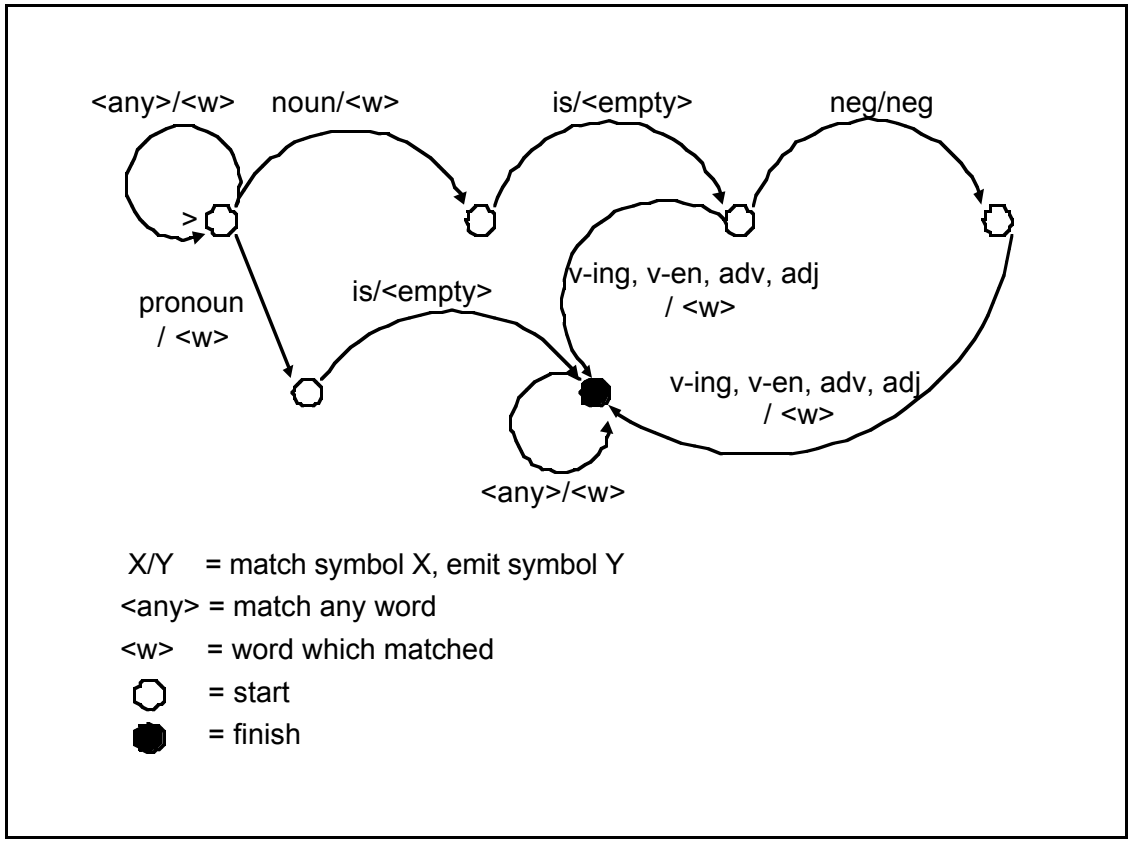

Figure 6. Example Finite State Transducer: Copula Deletion

qualitative changes (up, down, change, nochange) preceded by optional negation, and for parameters followed by qualitative changes. What is emitted is the "meaning" attribute from the lexical entry of the word (described later in this chapter), so that a word like "afterload" is converted to the concept map (for mean arterial pressure). Anything in the utterance which is not selected is deleted.

Negation. This machine looks for negation followed by a qualitative change and combines them. Thus an original utterance of "doesn't change" is transformed into neg + change in the parameter/change extraction step which becomes a unitary nochange in this step. There is no reason this machine needs to be restricted to negating qualitative changes, but there are no other cases for it right now.

Proportionality. This is another keyword extraction machine which looks for words and phrases meaning "direct" or "inverse." It is used for answers to questions about whether one parameter is directly or inversely proportional to another. Our medical students have a linguistic tic of using "indirect" as an antonym for the "direct" in "directly proportional."

Neural Mechanism. This machine is used for answers to the "by what mechanism is X controlled" question. It looks for an optional parameter name plus anything which can be matched to the mechanism ontology, shown in Figure 7. There are a number of near-miss answers in this ontology, like radius of arterioles and fiber length, which are not actually mechanisms of control. The mechanism ontology is described later in this chapter.

Easy Prepositional Phrases. These machines parse easy prepositional phrases which represent parameters. The central issue is that certain frequently occurring prepositional phrases have common meanings. For example the phrase "volume of blood," as it is normally used in our dialogues, can be taken to mean the volume of blood in the central venous compartment, which is the parameter cbv (central blood volume). It is tempting to put "volume of blood" in the lexicon with the meaning of cbv. However consider "volume of blood in the ventricle" which is equivalent to the parameter filling (ventricular filling). The phrase "volume in the ventricle" has the same meaning. The phrase "volume of blood ejected by the heart" means stroke volume, but it is beyond the capability of any transducer I have written. Similar phenomena occur (although maybe not as often) with "pressure" + {optional "of blood"} + preposition + anatomical part. Simple "pressure" means map but "pressure in the veins" means cvp. Finally there is a set of phrases beginning with words such as "length" or "radius" or "diameter" followed by an anatomical part, such as the ever-popular "radius of arterioles."

There have been two versions of these machines. One looks for all the possible patterns directly, and accepts the longest one which matches. It uses a small ontology to recognize the anatomical parts. "Volume," "pressure," "blood," "length," etc. are built into the machine, and the only prepositions it recognizes are "of" and "in." The second version adds an extra cascaded step which recognizes and replaces "blood volume" and "volume of {the} blood" before looking for the main prepositional phrase in the next machine.

Easy Student Initiatives. This is a transducer which runs before the question-specific cascades, looking for known easy student initiatives. Here is where the various forms of "I don't know" are recognized.

Hedges. Currently hedges are not used by CST, so this transducer has not been implemented. It will run early in every cascade, causing known hedges such as a question mark or "I think that" to be replaced by a hedge token. It could also check for easily recognizable inverted verbs, signaling an answer phrased as a question.

Equations. I have experimented with machines to recognize simple equations. From the prevalence of equations in the transcripts, I feel confident this capability will be needed in the future, but the current CST v. 2 asks no questions which routinely elicit equations from the students. Because of equations such as MAP = CO x TPR, we occasionally see student utterances such as "CO x TPR" as an answer to "what are the determinants of MAP?" Since the parameter-name extraction machine is simply looking for parameter names and ignoring the rest, such student answers are correctly recognized without recourse to a special recognizer for equations.

Lexicon and Lexical Lookup

The lexical lookup phase of the new input understander replaces a sentence by a list of lexical entry structures. This list is processed by the transducers.

The lexicon was derived from the words attested in the K-series keyboard-to-keyboard transcripts.[5] This list of words was edited and augmented by hand, inserting,

[5] Martha Evens, Kumar Ramachandran, and Murugan Kannan were the primary compilers and editors of the lexicon.

for example, phrases, synonyms, antonyms, abbreviations, and unattested forms. Each entry in the lexicon consists of a list of attribute-value pairs. Lexical attributes and values were manually derived, generally following the grammar of Quirk et al. [1972]. Here is a simple entry from the lexicon for the phrase "steady state:"

STEADY_STATE ((pos noun) (cm mass) (abbrev "ss") (mean ss))

"Steady state" is a mass noun. It is abbreviated by "ss" (which has its own entry in the lexicon) and it has a meaning of SS. Phrases are no different than other lexical entries. The individual components of a phrase, such as "steady" and "state," may or may not also have their own entries. Every distinct string has a separate entry, so that every form of a verb is entered separately, as are plurals, abbreviations, and so on. The different forms of a word are linked together, so that the inflected verbs are linked with the infinitive, the abbreviations and fully spelled-out forms are linked together, etc. Counting all distinct forms there are 4300 entries. Polysemous words increase the total number of entries to 5000.

There are many more attributes marked in the lexicon than are currently used by the input understander. The only attributes examined while the input understander is running are mean (meaning), pos (part-of-speech) and the lexeme itself. The cross-reference attributes such as those linking abbreviations with spelled-out forms, inflected with infinitive verb forms, and plurals with singular forms are processed off-line in the course of ensuring that the lexicon is consistent.

The different possible values for the mean attribute are shown in Table 6. Words have been assigned meanings only when needed for processing student input to answer CIRCSIM-Tutor's questions. Other words have no "meaning" assigned.

There is some polysemy in the lexicon, sometimes in less than obvious places. For example, the hyphen (which represents a minus sign) is in the lexicon as meaning both "decrease" and "inversely proportional." When there are multiple lexicon entries for the same word, all of them are fetched by lexical lookup.

When the finite state transducers are examining the next item in the sentence, that item is typically not a single lexical entry structure but (due to potential polysemy) a list of lexical structures. The transducer is usually looking for either a meaning attribute, a part-of-speech attribute, or the lexeme itself. In this case the transducer examines all the available lexical entry structures for that particular word, and makes a transition if any one of them matches. If the transducer emits a lexical entry during the transition, it will emit the one which matched, and the other polysemous senses are dropped.

The approach to phrases in the lexical lookup phase is what is colloquially known as the "maximal munch" strategy: the longest phrase in the lexicon which matches the next segment of input words is taken. There is a table of all proper prefixes of phrases in the lexicon to aid this strategy. A proper prefix is a sequence of words up to but not including the whole phrase: the proper prefixes of "right atrial pressure" are "right" and "right atrial."

Spelling Correction

Spelling correction is triggered in the lexical lookup phase only when a word or phrase is not recognized in the lexicon.

Table 6. Values of the MEAN Attribute in the Lexicon

MEAN	Meaning	MEAN	Meaning
ABV	arterial blood volume	F-S	Frank Starling effect
ANS	autonomic nervous system	HEDGE	<hedge>
BP	blood pressure	HR	heart rate
BR	baroreceptor	INVPROP	inversely
BRP	baroreceptor pressure	IS	inotropic state
BRR	baroreceptor reflex	KNOW	know/understand
BRRATE	baroreceptor firing rate	NEG	negation of verb: not, isn't, won't, etc.
BRSIZE	baroreceptor size	NO	no
BV	blood volume	NOCHANGE	unchanged
CBV	central blood volume	NS	nervous system
CC	cardiac contractility	PIT	intrathoracic pressure
CHANGE	changed	PHYSICAL	hemodynamic / non-neural control
CNS	central nervous systems	PNS	parasympathetic nervous system
CNSR	central nervous system response	PRELOAD	preload
CO	cardiac output	RA	arteriolar radius
CVP	central venous pressure	RAP	right atrial pressure
DIRPROP	directly	REFLEX	reflex
DN	decreased	RR	reflex response
DR	direct response	RV	venous resistance
DUNNO	doesn't know	SNS	sympathetic nervous system
EDP	end diastolic pressure	SS	steady state
EDV	end diastolic volume	SV	stroke volume
ESV	end systolic volume	TPR	total peripheral resistance
FILLING	ventricular filling	UP	increased
FL	fiber length	VR	venous return

Handling spelling errors has been a major function of the input understander in previous versions of CIRCSIM-Tutor [Lee 1990, Lee et al. 1990, Seu and Evens 1991, Seu 1992]. For the new input understander I am using the spelling corrector written by Mohammad Elmi [1994, Elmi and Evens 1998]. Its algorithm is an enhancement of the one from earlier CIRCSIM-Tutor spelling correctors.

The original CIRCSIM-Tutor spelling corrector by Lee was built around an algorithm which compared the misspelled word with every word in the lexicon. A sliding window selects several characters from the misspelled word and from the candidate word. These are then compared in three ways. If they are equal, the window slides over one, and proceeds.

If the sliding window is at character m_i of the misspelled word and c_j of the candidate word, Lee's algorithm compares m_i to c_j, m_i to c_{j+1}, and m_{i+1} to c_j.

The common one-character spelling errors produce distinctive mismatches in such a comparison. Single character substitutions, additions, and elisions are all identified and fixed up as the window slides over them. Multiple differences are thus fixed one at a time. Each fix-up is tabulated into a mismatch statistic, with a weight for each type of difference fixed. If the misspelled word and the candidate word are too different, as judged by the mismatch statistic, the candidate is rejected.

The output of the spelling correction algorithm is a list of words from the lexicon which approximately match the misspelled word, ranked according to degree of mismatch.

Elmi improved Lee's algorithm and analyzed its behavior thoroughly. He has shown that his four-way match algorithm (which additionally compares m_{i+1} to c_{j+1}) is

generally superior to previously published algorithms for correctly identifying candidates. Among other things, it is far superior at finding character reversals. Elmi also employs various techniques which are informed by observed categories of spelling errors. For instance, given that people are observed to abbreviate by shortening a word, letters which have been truncated from the end of the misspelled word are weighted less then letters elided from the middle. The mismatch computation is also adjusted by other factors, such as the observation that letters which have similar sounds are more likely to be confused, as are letters which are adjacent on the typewriter keyboard.

When a word in the student's utterance is not in the lexicon, the spelling corrector is called. The closest match from the lexicon is taken. In future versions of the input understander I hope to implement notions of topicality and locality as discussed in Chapter 9.

There is a special interaction between phrase lookup and spelling correction which in practice seems to improve the accuracy of correction considerably. If a word is not recognized, but the lexical lookup process is in the process of trying to match a phrase, the whole phrase prefix is spell-corrected. For example, the phrase "right atrial pressure" has a prefix "right atrial." Suppose the student's utterance contains "right atril." At the moment of spelling correction the input understander realizes it is trying to match a phrase, so it spell-corrects "right atril" instead of just "atril." I would like to extend this technique to phrase suffixes as well.

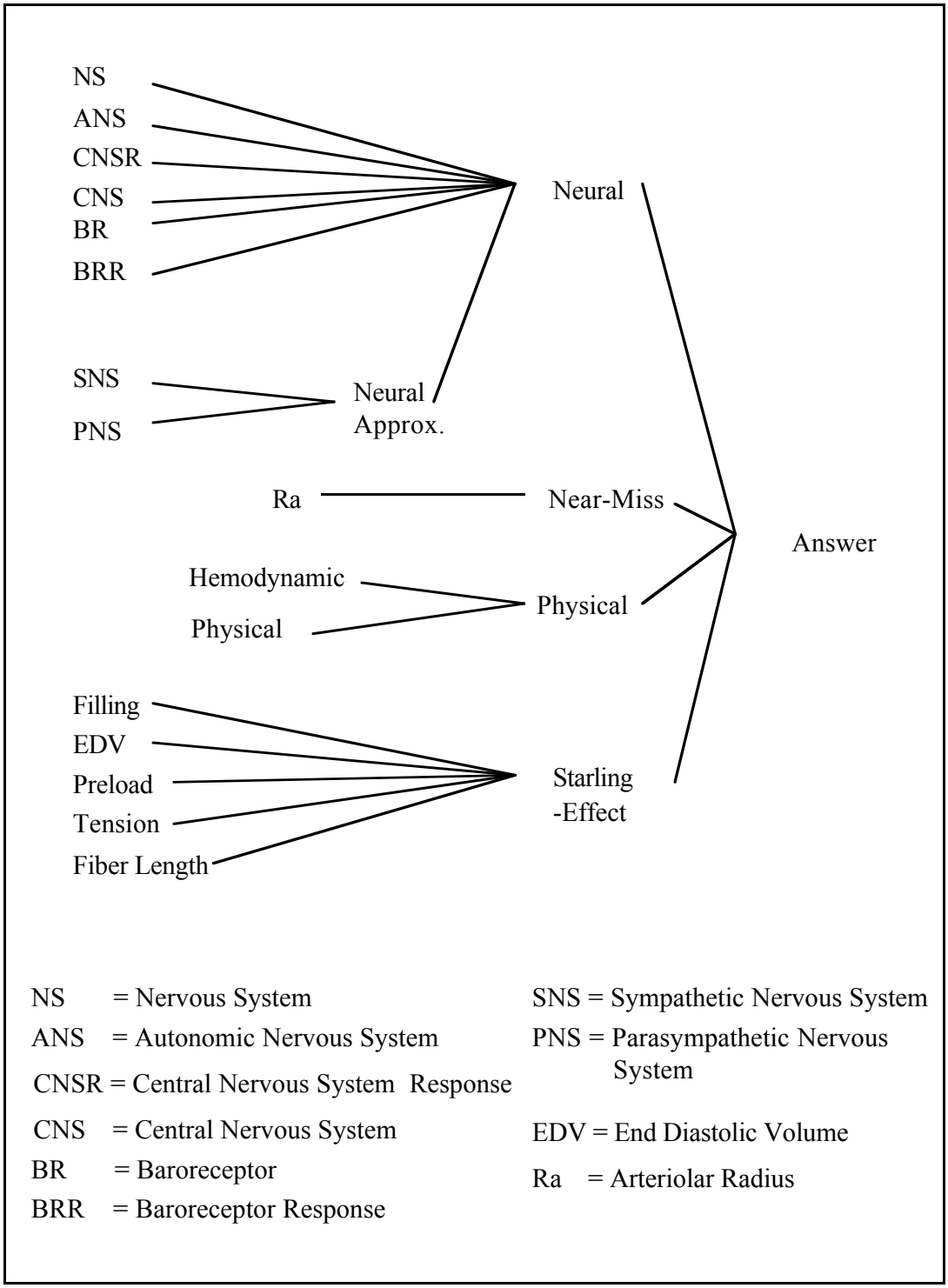

Figure 7. Mechanism Answer Ontology

Ontologies

The new input understander employs small ontologies during language processing. Primarily these are used by the transducers for matching symbols in the input stream to symbols needed for state-machine transitions. Thus, for example, a state machine might call for a transition based on scanning a parameter. Suppose the input is "arteriolar radius," which has a meaning attribute of ra. As part of symbol matching, the parameter ontology shows that ra is a parameter, so the match occurs.

The mechanism ontology is illustrated in Figure 7. It categorizes the kinds of meaning attributes for the answers to the question "by what mechanism is X controlled?" Arteriolar radius (ra) appears in this ontology (as well as in the parameter ontology), even though ra is properly speaking a parameter and not a mechanism of control. However ra can occur as an answer to the question "by what mechanism is TPR controlled," where it is categorized a near miss. Thus the mechanism ontology classifies ra as a kind of near miss.

Ontologies can be used by the input understander to classify words in ways other than "kind of answer." For example, the ontology for the recognition of prepositional phrases classifies various veins, arteries, heart chambers, and other anatomical terms as "containers," which is needed for understanding phrases such as "pressure in the central venous compartment."

Performance with a Class of Medical Students

When the first-year physiology class taught by Joel Michael and Allen Rovick at Rush Medical College used CIRCSIM-Tutor v. 2 in November, 1998, there were about fifty

students producing thirty-eight sessions (twenty-four of the students worked in pairs). In total, there were 1801 student turns.

Table 7 shows a summary of how many student turns were and were not recognized by the input understander. In this accounting, the ten garbled unusable turns include any turns for which I could not divine an intention of the student. All were no more than few characters long, for example the number "93," the letter "h," and the string

Table 7. Input Understander Performance in November 1998

Total student turns	1801
Garbled or ambiguous	10
Total usable turns	1791
Of total usable turns:	
Bare symbol (+/-/0)	393
Bare question mark	3
Alphabetic	1395
	1791
Of usable alphabetic turns:	
Recognized in a useful manner directly	1346
Not recognized in a useful manner	19
Recognized after correcting spelling, typing and impromptu abbreviations	30
	1395
Of not recognized alphabetic turns:	
Spelling or typing errors	6
Missing or incomplete lexical entry	5
Expressions of frustration	2
Abbreviations	2
"Help" and "OK"	2
Domain concepts beyond CST's knowledge	2
	19

"cvr." (It is possible that "cvr" stood for something in the mind of the student, but I couldn't divine it from context.) If there was a recognizable answer plus a few garbled characters additionally, the turn was counted as usable, not garbled.

Of the usable turns, some were simply the plus, minus, and zero symbols used to indicate values of increased, decreased, and no change. The input understander recognized all of these. I included the four times a student typed a single letter "o" instead of digit "0" in this category. Also there were three turns which consisted of a bare question mark. The question mark is recognized by the input understander, but it does nothing special with it, so the standard error message is emitted explaining what kind of answer is expected. I deemed this to be an appropriate response. Here is an example:

T: What parameter in the prediction table represents preload?
S: ?
T: Please respond with prediction table parameters
S: cvp
T: Right, Central Venous Pressure determines preload, therefore determines Stroke Volume.

The remaining 1395 turns were almost entirely alphabetic characters. Of these, nineteen (1.4% of alphabetic turns, 1.1% of all usable turns) were not recognized by the input understander. Table 8 shows examples of unrecognized input. Of the turns which were recognized, thirty required spelling correction. Representative examples are shown in Table 9.

It should be noted that in addition to the input understander phenomena described above, many of the tutoring phenomena we had planned for occurred during this session. Students produced many near-miss answers and exercised many of the new hints that CIRCSIM-Tutor v. 2.5 is capable of producing.

Table 8. Representative Samples of Input Understander Failure in November 1998

Student Input	Reason Not Recognized
"increaseed"	Spelling correction algorithm did not repair it.
"istpr"	Here "is" and "tpr" are joined, but joins are not currently recognized by spelling correction.
"neurological"	There was a defect in the lexicon entry for this word.
"central venous volume"	Missing from the lexicon
"kiss my ass"	Expressions of frustration are not explicitly recognized.
"in"	This is a too-drastic abbreviation for "inverse."
"help"	Requests for help are not currently recognized as a category, thus the word "help" in the lexicon has no assigned meaning.
"metabolic factors"	This is a domain concept beyond CST's knowledge, so the words have no assigned meaning in the lexicon

Table 9. Representative Spelling Errors Corrected in November 1998

Student Input	Result of Correction
"rh"	"hr"
"trp"	"tpr"
"cariac"	"cardiac"
"inotrphic"	"inotropic"
"troke"	"stroke"
"cahnge"	"change"

CHAPTER IX

MORE LANGUAGE-ORIENTED TOPICS FOR INPUT UNDERSTANDING

Spelling Correction and the Input Understander

It is my suggestion that the CIRCSIM-Tutor input understander can further improve its spelling correction by using notions of locality and topicality to pick from possible candidate corrections. This simple idea doesn't seem to occur in most spelling correction software, but should make a noticeable difference.

Here is a short extract from transcript K12 containing two problems which I will use to illustrate the idea. Please note "immed." and "ventricale" in the following:

T: Does venous return go up immediately?
S: Does the rate of blood removal from the central veins mean that blood entering the right atrium, if so i think venous return does go up immed.
T: We need to get our terminology straight. Venous return means blood returning from the systemic circulation to the heart. That does not go up immediately. It takes about a minute after CO I. Does more blood enter the ventricale for CO to I, Yes. [K12-45]

The abbreviation "immed." in this example occurs only once in our eighty or so transcripts. It happens shortly after the use of the spelled-out word "immediately." I suggest that this is not a coincidence, that it is natural for impromptu abbreviations to occur on second and subsequent uses. Let us call this "locality."

The misspelling "ventricale" for "ventricle" is analyzed a little differently. Ventricles have not been mentioned in the preceding dialogue. However central veins and the right atrium have been mentioned. This is a discussion which includes anatomy. In a discussion of anatomy, ventricle fits right in. Let us call this "topicality."

It may be objected that this sample discourse is far more complicated than anything the new input understander will need to process. This objection does not invalidate my claim that locality and topicality might be useful. Even in the realm of short answers it may be that CIRCSIM-Tutor will ask a question for which "ventricle" is a plausible answer. For example, imagine the following hypothetical exchange:

T: What determines CO?
S: Ventricale filling.

The spelling correction algorithm might propose both "vertical" and "ventricle" from the lexicon. One of the two must be picked. There may possibly be several methods for picking. I am proposing that if there had been a discussion of cardiac anatomy immediately preceding, the input understander could keep track of that fact and use it to pick "ventricle."

It might be possible for the input understander to implement a notion of locality which is rather more sophisticated than simply "within the previous *k* words" for some constant *k*. In particular, I would like to try something approximating "within the currently open discourse (or pedagogical?) segments."

The key here is that the tutor is controlling the conversation. Within one problem, the tutor first gives the student the problem description, then prompts for the student's predictions, then tutors the errors. Within the tutoring phase, there is a conversation segment for each of the three stages of the problem. Within each stage, there is a segment for each incorrect variable [Freedman 1996, Freedman and Evens 1996]. While correcting

each variable, there is a segment for each attempt to remediate the error. Each of these segments of conversation is derived from a separate goal in the tutorial planner.

At most points in time, it should be possible to examine the tutorial planner's current plan tree and identify the open discourse segments. Then the input understander can take the vocabulary which exists in the discourse history and build a block structured table of recently used words, those in the currently open discourse segments, to use whenever it invokes locality to disambiguate a word.

Implementing the notion of topicality will start with the notion of locality and add an appeal to the concept ontology. In the above transcript example, where "ventricle" is correct and "vertical" isn't, several anatomical terms were mentioned in the current conversation segment, in particular "atrium." Both "atrium" and "ventricle" are specializations of "cardiovascular-part" in the ontology. The input understander can examine each word in the locality table, looking it up in the concept ontology, to see if it has any close relations to the words proposed by the spelling corrector.

It must be noted that identifying what I call "topicality" here is a small part of what has been done elsewhere. In a seminal paper, Grosz [1977] presents an algorithm to dynamically identify the "focus space" of a conversation, represented as a partition of a semantic network. If we had more complex dialogue, this kind of analysis would be what is needed.

Use of Equations in the Transcripts

I decided to study equations in the transcripts with several goals in mind. First, it would be useful to describe the language involved if we are to ever generate or understand

text containing equations. Next, I wanted to see how equations might challenge the CIRCSIM-Tutor v. 3 input understander, even given the limited conception of the kinds of questions v. 3 might ask. We have attested examples of students using such equations in response to simple questions. Finally, since none of the common tutoring tactics and schemas described in earlier work for CIRCSIM-Tutor involve equations, I wanted to see what new tactics and schemas I could find.

There are only two equations which occur with regularity in the tutoring transcripts: $CO = SV \times HR$ and $MAP = CO \times TPR$. Here I catalog the contexts in which they occur.

I have extracted 45 instances among the 50 K-transcripts and 36 instances among the 31 N-transcripts where one or the other of the above equations was mentioned. An "instance" may contain several occurrences of the same equation, for example when the student is trying to understand it or states it incorrectly, or somebody repeats back an equation just introduced, so the total number of occurrences of equations in the text is somewhat higher. I located equations in the text by searching for strings as shown in Table 10.

Table 10. Identifying Equations

String	Typical Occurrence
" x "	CO x TPR
"*"	CO * TPR
"times"	sv times hr
"prod"	MAP is the product of TPR and CO
"="	MAP = …
"equa"	"equation" in textual proximity to an equation

The equations in the transcripts almost always conform to the following grammar. In the equation "MAP = CO x TPR" I refer to MAP as the LHS and "CO x TPR" as the RHS. If you write the equation in reverse as "CO x TPR = MAP" I still refer to the single variable side as the LHS.

LHS → variable
RHS → variable MULT-SYMBOL variable |
"the product of" variable "and" variable
EQN → LHS EQ-SYMBOL RHS | RHS EQ-SYMBOL LHS
MULT-SYMBOL → "*" | "x" | "times" | space | null-string
EQ-SYMBOL → "=" | "is" | "is equal to" | other verbs

Note that some of the verbs which can be used for EQ-SYMBOL are not symmetric predicates of equality. For example if you use "determines," the sentence "HR times CO determines TPR." would not mean the same as "TPR determines HR times CO."

Equations can occur in isolation, or they can occur as constituents of sentences. They can be expressed in their entirety or only the RHS might be expressed. All four cases, isolated vs. run-in and entire vs. RHS only, are attested:

(1) CO=TPR*HR [K3-st-22-1]
(2) But isn't CO X TPR =MAP ? [K7-st-100-1]
(3) SV X HR [K38-st-49-1]
(4) Co does decr (the product of sv and hr is down)
and tpr is decr. [K25-tu-154-2]

It is rare to find an RHS which occurs by itself as a constituent of a sentence as in (4). Usually the corresponding LHS is also present, joined by a verb or an equal sign. However once an equation has been embedded into a sentence it is subject to the normal syntactic transformations. In (5) we have an example of an equation run into a sentence where the LHS "MAP" has been pronominalized to "it." This shows clearly that the LHS and RHS are distinct syntactic constituents:

(5) I did forget to tell you that MAP increased because it is equal to CO times TPR. [N5-tu-44-1]

In (6) below, in addition to LHS pronominalization, the equation has been passivized around the EQ-SYMBOL "determines," thereby moving LHS and RHS in relation to each other:

(6) This caused the CO to decrease because it is determined by SV times HR, but HR does not change. [N7-tu-66-10]

There is a strong preference for using symbols over words; "times" and "equals" are rather uncommon compared to "*" or "x" and "=". It would be tempting, from a parsing point of view, to separate equations into two varieties, one using symbols and one using words. There are a few equations which have no symbols. Here is another:

(7) The cardiac output equals stroke volume times heart rate. [K9-st-24-2]

However, hybrid examples such as the following motivate one to produce a single combined grammar:

(8) Sv times hr =co [K13-st-52-1]

Examples (5) and (6) demonstrate that when equations are embedded into a sentence, the LHS "it" and the RHS "SV times HR" or "CO times TPR" are syntactically serving as NPs. If we were to parse equations which are embedded in sentences, I would argue for treating "times" and the other multiplication symbols as a kind of conjunction between two NPs.

A dramatic example of what can be interpreted as an equation run into a sentence followed by a syntactic transformation is:

(9) But isn't CO X TPR =MAP? [K7-st-100-1]

This would seem to argue that EQ-SYMBOL is serving as a verb.

Note, however, that most often an equation run into a sentence is treated as a clause unto itself, even sometimes an independent clause or an embedded S:

(10) MAP = CO X TPR is a deterministic statement. [K40-tu-108-2]
(11) SV x HR = CO so the cardiac output will incise also. [K40-st-65-1]
(12) …Since HR doesn't change in the DR, CO must have gone up (CO = HR X SV). [K39-tu-78-6]

Note that when the equation is expressed in symbols, inter-word blanks are often not written, e.g. "CO=TPR*HR" instead of "CO = TPR * HR." Usually this is not a problem, since symbols are not normally part of variable names or English words in general. For example, the blank missing in "=MAP" in example (9) should cause little difficulty. However when "x" is used as a multiplication symbol the result of missing blanks is mysterious words, e.g. "coxtpr" (which occurs five times in the transcripts), "svxhr," and "hrx" (from "hrx sv"), among other attested examples.

An occasional construction is the elision of the multiplication symbol, mimicking common mathematical practice. The result is not always pretty because the two operands are sometimes combined into one word without a space:

(13) CO = HR SV [K3-tu-23-2]
(14) Co=hrsv [K25-st-65-1]

It seems that if we are to accept equations as student input a certain number of *ad hoc* lexical entries and grammar rules might be convenient. We can put all the combinations such as "svxhr" and "svhr" into the lexicon, perhaps as abbreviations. The scanner will separate symbols from adjoining words, so that "TPR*HR" will become

three tokens. And the grammar will have to recognize an equation as a possibly independent clause which can be mixed with the other constituents.

As I have mentioned previously, students sometimes really do type equations in response to questions which do not ask for one. Here are two examples:

(15) T: When CO increases, does it affect the value of another variable?
S: Yes, MAP=CO * TPR. [K4-tu-41]

(16) T: But, what are the determinants of co?
S: Hr x sv [K42-tu-110]

My feeling is that these cases are ones that CIRCSIM-Tutor v. 3 truly ought to understand. It should be able to deduce the expected answers, "MAP" and "CO, TPR" respectively, from the student's response.

How the equation is interpreted will depend on the question which was asked. Let me suggest how to process example (15):

- Recognize that the input is an equation MAP = CO * TPR, build an internal representation for it.
- Verify that the original question, "what variables does parameter X affect," is one that is known to be answerable by an equation. Locate the method for doing so, expressed in the next few steps.
- Verify that the parameter in the question, CO, is on the right hand side of the student's equation.
- Find the left hand side, if one was input by the student, and propose this as the answer. In this case that is MAP.

- As a side issue, verify that the equation is correct. If the student typed, for example, "MAP = CO * SV," which is false, we still have a correct answer "MAP is affected by CO." However the erroneous equation should not pass by without comment; CIRCSIM-Tutor might want to emit some sort of correction. How to do that is up to the instructional planner.
- At this point, processing can continue as if the student had typed "MAP"

Note that processing (16) will contain some different steps in equation interpretation, because the original question was different:

- The original question is "what parameters determine variable X."
- Verify that the parameter in question, CO is on the left hand side of the equation. If the student didn't specify any LHS we can assume it is CO.
- Find the parameters on the right hand side as input by the student, and propose these as the answer.
- Again, we verify that the equation is correct so a correction could be issued if needed.

Note that to process (15) we extracted the answer from the left hand side, and to process (16) we extracted the answer from the right hand side.

One important thing to notice here is that there can be several results which simultaneously come from processing the student's answer. In addition to the straight answer to the question, there is the side issue of whether the equation was correct. This side issue was not among the tutor's current goals, but if the student types the equation incorrectly the student introduces a new tutoring goal to correct the equation (albeit a goal

which will be only briefly addressed). For this reason, it might be processed in the same manner as a simple student initiative.

There are several tutoring schemas involving equations which appear in the expert tutor transcripts. Here are two which occur more than once and are clear enough for me to abstract the logic.

A frequent goal which causes the tutor to introduce the equation CO = HR x SV is to teach that HR is (in this case) a stronger determinant of CO than SV is. The strategy is to point out that:

- HR has increased
- SV has decreased
- CO = HR x SV means that HR and SV are influencing CO in opposite directions
- CO increased, so HR is the stronger determinant

Sometimes the tutor elicits the equation from the student then the tutor supplies the parameter values, sometimes the tutor states the equation and lets the student discover the conflict. It is possible for the tutor to provide everything or to elicit all steps from the student in a directed line of reasoning. In any event, tutoring this goal often involves invoking an equation. Here is one of the shorter examples of this type:

(17) T: What two variables determine CO?
S: HR and SV
T: Right. CO = HR x SV. So, as long as HR increases more than SV decreases, CO will be increased. [K49-tu-62]

Another situation which occurs several times in the expert transcripts is to teach the student the value of CO in steady state, given already known values for MAP and TPR. The reasoning is:

- TPR has increased (for instance)
- MAP has decreased
- MAP = CO x TPR, so CO must have decreased MAP more than TPR increased it.

I note parenthetically that the reason this goal occurs in steady state is that the tutors usually do not try to reason causally among the steady state values. In DR and RR, the predictions were derived from the procedure variable, causal relations among the concept map variables, and various principles such as "neural variables do not change in DR." In SS the tutors usually derive their predictions from the DR and RR values, which enables them to obtain values for TPR and MAP with some believability. The value for CO can be derived by a consistency argument which invokes this equation.

Here is a brief example of that kind of schema. As with other tutoring schemas, the component propositions can be either elicited from the student or contributed by the tutor.

(18) T: Well, you previously told me that map=co x tpr.
You predicted that map was still i and tpr MUST be d since it was 0 in dr and d in rr.
So what about co? [K44-tu-206]

Sometimes the expert tutors try to elicit equations from the student as a last resort to get determinants after some other tactic has failed. For example, in K39 the tutor writes "Write an equation using only the variables in the prediction table that says MAP =," but

the tutor isn't actually interested in obtaining that equation. It is the tutor's third and final attempt to elicit from the student the determinants of MAP, which would have been evident had the student responded with the correct equation.

A common locution is to interpolate an equation to support a causal argument. Similarly, students often invoke equations when asked to explain their reasoning. Here are some examples:

(19) A fall in TPR would cause a fall in MAP (map=TPR*co) [K3-st-64]

(20) T: Next?
S: CO
T: Change?
S: I, as HR X SV= CO [K39-tu-30]

(21) T: In what way can co affect sv?
S: Co=hr x sv so an increased co would mean an increased sv [K4-tu-39]

(22) T: In this case the first thing that changed was HR- it increased. This caused CO to increase because of CO= HRxSV. Now because CO goes up, so does MAP (because of MAP= CO x TPR). [N12-tu-52]

Example (21) contains a misconception about using equations to support causal arguments which occurs from time to time in student thinking. In this domain, physiological causality proceeds from the right hand side determinants to the left hand side result.

A final use of equations in service of an ancillary goal is to invoke an equation to support a statement that one parameter is a determinant of another.

(23) S: I would say that TPR predicts it because that controls the pressure.
T: So does the CO control the pressure, MAP = CO X TPR. [K39-st-175]

(24) T: Ok, remember that MAP is dependent on CO and TPR (MAP =TPR x CO) [N2-tu-110]

CHAPTER X

CONCLUSIONS

Summary and Significance

CIRCSIM-Tutor is among a small number of intelligent tutoring systems where the tutor has control of a dialogue and the student types free text. That this modality is rare in a tutoring system might be surprising, since human tutors commonly let their students talk almost at will. But being able to comprehend and act on the freely expressed thoughts of a student is a formidable task which almost nobody has undertaken. Nevertheless Joel Michael and Allen Rovick, authors and users of several more traditional tutoring systems, perceived the desirability of a tutoring system which at some level engages in a dialogue with the student.

One of the goals of this thesis is therefore to justify that perception: to show that CIRCSIM-Tutor's mode of tutoring works, that tutoring works in general, that asking questions and having students utter answers and explanations increases their learning. As part of that justification I compared expert to novice tutors and showed that the experts, in one specific way, cause students to generate (rather than listen to) more answers than the novice tutors do.

CIRCSIM-Tutor severely constrains the range of student utterances by asking very closed questions and by responding to only a limited variety of responses. I have managed to show from human tutoring transcripts and from logs of previous versions of CIRCSIM-Tutor that even within this constrained domain there are a number of phenomena

that can be exploited and a number of phenomena which must be coped with. I have done a extensive analysis of the kind of language which students use in these tutoring sessions.

Among the phenomena that we can now or may soon be able to exploit are "near miss" answers, where the student says something unexpected but still usable for tutoring, answers which are known to be diagnostic of various misconceptions, equations interpolated into the answers, and answers which are hedged.

I examined some of the available technologies for input understanding, and wrote a new input understander for CIRCSIM-Tutor. The new understander is based on the technique of finite state transducers, using the robust approach of extracting only what is needed from the student answer and ignoring the rest. Having a new understander helped to upgrade the tutoring ability of CST v. 2, which is now being used by medical students. Furthermore the new input understander will be able to handle the kinds of student input we expect in the new CST v. 3, which is currently being written by other members of our group.

I have performed experiments with the new technology of Latent Semantic Analysis. I discussed how CIRCSIM-Tutor might use the technique to sensibly handle student answers to "diagnostic questions," a class of much more open questions than CST now asks.

Engaging in dialogue, particularly tutorial dialogue, is at the forefront of research right now. Endeavors like the Human Tutorial Dialogue Project [Fox 1993] are studying how tutorial dialogue works and how it might be adapted to computers. Other tutoring projects are starting to realize the potential for free-text student input, and a recent one

(AutoTutor) has plunged into that area in ways not tried before. The new input understander enables CIRCSIM-Tutor to go into new territory and advance the state of the art in intelligent tutoring. I am proud to be a part of that effort.

Future Work

There is considerable future work to be done in input understanding for CIRCSIM-Tutor. There are phenomena which it is ready to handle that I have not implemented fully, such as hedges and equations. Furthermore, as the current tutor improves, the input understander needs to be upgraded to match its needs.

There is an experiment in using Latent Semantic Analysis to process diagnostic questions which I hope to perform at Rush Medical College the spring of 1999. If the experiment works, it may be a big improvement for CIRCSIM-Tutor.

Finally, CIRCSIM-Tutor v. 3 is on the horizon; parts of it are being written right now. The new input understander has the technology and the structure to meet the needs of the new tutor, and to provide it with access to student answer phenomena which have not been previously available.

BIBLIOGRAPHY

Aleven, Vincent, Kenneth R. Koedinger, H. Colleen Sinclair and Jaclyn Snyder. 1998. "Combatting Shallow Learning in a Tutor for Geometry Problem Solving." In Goettl et al., 1998, pp. 364–373.

Anderson, John R. 1993. *Rules of the Mind*, Hillsdale, NJ, Lawrence Erlbaum Associates.

Anderson, John R., Albert T. Corbett, Kenneth R. Koedinger and Ray Pelletier. 1995. "Cognitive Tutors: Lessons Learned," *Journal of the Learning Sciences*, vol. 4, pp. 167–207.

Anderson, Richard C. and William E. Nagy. 1993. *The Vocabulary Conundrum*. Technical Report 570 from the Center for the Study of Reading, College of Education, University of Illinois, Urbana, IL. Available from ERIC, document number ED354489.

Anwar, Ashraf, Fergus Nolan, Melissa Ring, Katja Wiemer-Hastings and Peter Wiemer-Hastings. 1998. *Report on the Status of LSA as used in AutoTutor*, unpublished status report of the Tutoring Research Group, Departments of Psychology and Mathematical Sciences, University of Memphis, Memphis, TN.

Berry, Michael, Theresa Do, Gavin O'Brien, Vijay Krishna and Sowmini Varadhan. 1993. *SVDPACKC (Version 1.0) User's Guide*, Report CS-93-194, Computer Science Department, University of Tennessee at Knoxville. Revised 1996.

Berry, Michael, Susan T. Dumais and Gavin O'Brien. 1994. *Using Linear Algebra for Intelligent Information Retrieval*, Report CS-94-270, Computer Science Department, University of Tennessee at Knoxville.

Bloom, Benjamin. 1984. "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring," *Educational Researcher*, vol. 13, no. 6, pp. 4–16.

Bresnan, Joan and Ronald Kaplan. 1982. "Lexical-Functional Grammar: A Formal System for Grammatical Representation," in Bresnan, Joan, ed., *The Mental Representation of Grammatical Relations*, Cambridge, MIT Press, pp. 173–281.

Brown, John Seely and Richard R. Burton. 1975. "Multiple Representations of Knowledge," in Bobrow, Daniel G., *Representation and Understanding: Studies in Cognitive Science*, New York, Academic Press, pp. 311–349.

Burton, Richard R. and John Seely Brown. 1979. "Toward a Natural Language Capability for Computer-Assisted Instruction," in O'Neil, H., ed., *Procedure for Instructional Systems Development,* New York, Academic Press, pp. 272–313. Reprinted in Grosz, Barbara et al., eds., *Readings in Natural Language Understanding*, San Francisco, Morgan Kaufmann, 1986, pp. 605–625.

Cawsey, Alison. 1992. *Explanation and Interaction*, Cambridge, MA, MIT Press.

Chi, Michelene T. H., Nicholas de Leeuw, Mei-Hung Chiu and Christian LaVancher. 1994. "Eliciting Self-Explanations Improves Understanding," *Cognitive Science*, vol. 18, pp. 439–477.

Cohen, P., J. Kulik and C. Kulik. 1982. "Educational Outcomes of Tutoring: A Meta-Analysis of Findings," *American Educational Research Journal*, vol. 19, pp. 237–248.

Coleman, Elaine B., Ann L. Brown and Inna D. Rivkin. 1997. "The Effect of Instructional Explanations on Learning from Scientific Texts," *Journal of the Learning Sciences*, vol. 6, no. 4, pp. 347–366.

Corbett, Albert T., Holly J. Trask, Christine Scarpinatto and William S. Hadley. 1998. "A Formative Evaluation of the PACT Algebra II Tutor: Support for Simple Hierarchical Reasoning." In Goettl et al., 1998, pp. 374–383.

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer and Richard Harshman. 1990. "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, vol. 42, pp. 391–407.

Dickinson, C. J., C. H. Goldsmith and D. L. Sackett. 1973. "MACMAN: A Digital Computer Model for Teaching Some Basic Principles of Hemodynamics," *Journal of Clinical Computing*, vol. 2, no. 4, pp. 42–50.

Dumais, Susan T. 1991. "Improving the retrieval of information from external sources," *Behavior Research Methods, Instruments and Computers*, vol. 23, no. 2, pp. 229–236.

Edelson, Daniel C. 1996. "Learning from Cases and Questions: The Socratic Case-Based Teaching Architecture," *Journal of the Learning Sciences*, vol. 5, no. 4, pp. 357–410.

Elmi, Mohammad Ali. 1994. *A Natural Language Parser with Interleaved Spelling Correction Supporting Lexical Functional Grammar and Ill-Formed Input*, Ph.D. dissertation, Illinois Institute of Technology, Chicago.

Elmi, Mohammad Ali and Martha W. Evens. 1998. "Spelling Correction using Context," *Proceedings of the 17th International Conference on Computational Linguistics (COLING–98), Montreal*, also known as *36th Annual Meeting of the Association for Computational Linguistics (ACL '98)*, pp. 360–364.

Fox, Barbara A. 1993. *The Human Tutorial Dialogue Project*, Hillsdale, NJ, Lawrence Erlbaum Associates.

Frasson, Claude, Gilles Gauthier and Alan Lesgold, eds. 1996. *Intelligent Tutoring Systems: Third International Conference, ITS '96, Montreal, Canada, June 1996.* Berlin, Springer. Published in the series *Lecture Notes in Computer Science*, vol. 1086.

Frederking, Robert E. 1988. *Integrated Natural Language Dialogue: A Computational Model*, Boston, Kluwer.

Freedman, Reva. 1996. *Interaction of Discourse Planning, Instructional Planning and Dialogue Management in an Interactive Tutoring System*, Ph.D. dissertation, Northwestern University, Evanston.

Freedman, Reva and Martha W. Evens. 1996. "Generating and Revising Hierarchical Multi-Turn Text Plans in an ITS." In Frasson et al., 1996, pp. 632–640.

Freedman, Reva, Yujian Zhou, Michael Glass, Jung Hee Kim and Martha W. Evens. 1998a. "Using Rule Induction to Assist In Rule Construction for a Natural-Language Based Intelligent Tutoring System," *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society, Madison, WI*, Hillsdale, NJ: Lawrence Erlbaum, pp. 362–367.

Freedman, Reva, Yujian Zhou, Jung Hee Kim, Michael Glass and Martha W. Evens. 1998b. "SGML-Based Markup as a Step toward Improving Knowledge Acquisition for Text Generation," *AAAI 1998 Spring Symposium: Applying Machine Learning to Discourse Processing*, Stanford, pp. 114–117.

Gertner, Abigail S. 1998. "Providing Feedback to Equation Entries in an Intelligent Tutoring System for Physics." In Goettl et al., 1998, pp. 254–263.

Glass, Michael. 1986. *Evaluating the Online Catalog*, M.S. thesis, Illinois Institute of Technology, Chicago.

Glass, Michael. 1997. "Some Phenomena Handled by the CIRCSIM-Tutor v. 3 Input Understander," *Proceedings of the 10th International Florida Artificial Intelligence Symposium (FLAIRS '97)*, Daytona Beach, FL, pp. 21–25.

Glass, Michael and Martha W. Evens. 1996. "Goals for the CIRCSIM-Tutor v. 3 Input Understander," in M. Gasser, ed., *Online Proceedings of the 1996 Midwest Artificial Intelligence and Cognitive Science Conference*, URL http://www.cs.indiana.edu/event/maics96/Proceedings/glass.html.

Goettl, Barry, Henry Halff, Carol Redfield and Valerie Shute, eds. 1998. *Intelligent Tutoring Systems: 4th International Conference (ITS '98) San Antonio, TX*, Berlin, Springer. Published in the series *Lecture Notes in Computer Science*, vol. 1452.

Graesser, Art. 1993. *Questioning Mechanisms During Tutoring, Conversation, and Human-Computer Interaction*, Technical Report R&T 4422576 of the Cognitive Science Program, Office of Naval Research.

Graesser, Art, Stan Franklin, Peter Wiemer-Hastings, and the Tutoring Research Group. 1998. "Simulating Smooth Tutorial Dialogue with Pedagogical Value," *Proceedings of the 11th International Florida Artificial Intelligence Symposium, Sanibel Island, FL*, pp. 163–167.

Granger, Richard H. 1983. "The NOMAD System: Expectation-Based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-Formed Text," *American Journal of Computational Linguistics*, vol. 3, nos. 3–4 (1983), pp. 188–196.

Grosz, Barbara. 1977. "The Representation and Use of Focus in a System for Understanding Dialogs," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA*, pp. 67–76. Reprinted in Grosz, Barbara et al., eds., *Readings in Natural Language Understanding*, San Francisco, Morgan Kaufmann, 1986, pp. 605–625.

Hirschman, Lynette and Marc Vilain. 1995. *ACL '95 Tutorial on Extracting Information from the MUC*, Bedford, Mass., Mitre Corp.

Hobbs, Jerry R., Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel and Mabry Tyson. 1997. "FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text," in Roche and Schabes, 1997, pp. 383–406.

Hopcroft, John E. and Jeffrey Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*, Reading, MA: Addison-Wesley.

Hume, Gregory. 1995. *Using Student Modeling to Determine When and How to Hint in an Intelligent Tutoring System*, Ph.D. dissertation, Illinois Institute of Technology, Chicago.

Khuwaja, Ramzan Ali. 1994. *A Model of Tutoring: Facilitating Knowledge Integration Using Multiple Models of the Domain*, Ph.D. dissertation, Illinois Institute of Technology, Chicago.

Kim, Jung Hee, Reva Freedman, and Martha W. Evens. 1998. "Relationship Between Tutorial Goals and Sentence Structure in a Corpus of Tutoring Transcripts," *Ninth Midwest AI and Cognitive Science Conference, Dayton, OH*, Menlo Park, CA: AAAI Press, pp. 124–131.

Landauer, Thomas K., Peter W. Foltz and Darrell Laham. 1998. "An Introduction to Latent Semantic Analysis," *Discourse Processes*, vol. 25, pp. 259–284.

Landauer, Thomas K. and Susan T. Dumais. 1997. "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge," *Psychological Review*, vol. 104, pp. 211–240.

Lavie, Alon. 1996. GLR*: A Robust Grammar-Focused Parser for Spontaneously Spoken Language, Ph.D. dissertation, Carnegie Mellon University, Pittsburgh.

Lee, Yoon-Hee. 1990. *Handling Ill-Formed Natural Language Input for an Intelligent Tutoring System*, Ph.D. dissertation, Illinois Institute of Technology, Chicago.

Lee, Yoon-Hee, Martha Evens, Joel A. Michael and Allen A. Rovick. 1990. "Spelling Correction for an Intelligent Tutoring System," *Proceedings of the First Great Lakes Computer Science Conference, Kalamazoo, MI,* Springer. Published in the series Lecture Notes in Computer Science, vol. 507.

Mark, Mary A., Kenneth R. Koedinger and William S. Hadley. 1998. "Élaborating Models of Algebraic Expression-Writing." In Goettl et al., 1998, pp. 524–533.

Martin, Charles. 1990. *Direct Memory Access Parsing*, Ph.D. thesis, Yale University, New Haven. Available as Technical Report CS 93-07 from the University of Chicago Department of Computer Science, 1993.

Mollá Aliod, Diego, Jawad Berri and Michael Hess. 1998. "A Real World Implementation of Answer Extraction," *Proceedings of the 9th International Conference and Workshop on Database and Expert Systems*, Workshop on Natural Language and Information Systems (NLIS '98), Vienna. pp. 143–148.

Moore, Johanna D. 1995. *Participating in Explanatory Dialogues*. Cambridge, MA, MIT Press.

MUC. 1993. *Fifth Message Understanding Conference (MUC–5): Proceedings of a Conference held in Baltimore, Maryland, August 25–27, 1993*, San Francisco, CA, Morgan Kaufmann.

MUC. 1995. *Sixth Message Understanding Conference (MUC–6): Proceedings of a Conference held in Columbia, Maryland, November 6–8, 1995*, San Francisco, CA, Morgan Kaufmann.

MUC. 1998. *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference held in Fairfax, Virginia, April 29–May 1, 1998*, SIAC Corp. Published online at URL: http://www.muc.saic.com/proceedings/muc_7_proceedings/overview.html.

Nagy, William E. 1995. *On the Role of Context in First- and Second-Language Vocabulary Learning*, Technical Report 627 from the Center for the Study of Reading, College of Education, University of Illinois, Urbana, IL. Available from ERIC, document number ED391152.

Nagy, William E. and Richard C. Anderson. 1984. "How Many Words are there in Printed School English?," *Reading Research Quarterly*, vol. 19, no. 3, pp. 304–330.

Pinker, Steven. 1994. *The Language Instinct*. New York, William Morrow and Company.

Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1972. *A Grammar of Contemporary English*. London, Longman.

Riesbeck, Christopher and Roger Schank. 1989. *Inside Case-Based Reasoning*, Hillsdale, NJ, Lawrence Erlbaum Associates.

Roche, Emmanuel and Yves Schabes. 1997. *Finite-State Language Processing*, Cambridge, MA, MIT Press.

Rovick, Allen A. and Lisa Brenner. 1983. "HEARTSIM: A Cardiovascular Simulation with Didactic Feedback," *Physiologist*, vol. 26, no. 4, pp. 236–239.

Rovick, Allen A. and Joel A. Michael. 1986. "CIRCSIM: IBM-PC Computer Teaching Exercise on Blood Pressure Regulation," *Proceedings XXX Congress of International Union of Physiological Sciences*, Vancouver, p. 138. Poster abstract.

Rovick, Allen A. and Joel A. Michael. 1992. "The Predictions Table: a Tool for Assessing Students' Knowledge," *American Journal of Physiology*, vol. 263, no. 6, pt. 3, pp. S33–S66. Also available as *Advances in Physiology Education*, vol. 8, no. 1, pp. S33–S36.

Sanders, Gregory A. 1995. *Generation of Explanations and Multi-Turn Discourse Structures in Tutorial Dialogue, Based on Transcript Analysis*, Ph.D. dissertation, Illinois Institute of Technology, Chicago.

Seu, Jai Hyun. 1992. *The Development of an Input Understander for an Intelligent Tutoring System Based on a Sublanguage Study*, Ph.D. dissertation, Illinois Institute of Technology, Chicago.

Seu, Jai, Ru-Charn Chang, Jun Li, Martha W. Evens, Joel A. Michael and Allen A. Rovick. 1991. "Language Differences in Face-to-Face and Keyboard-to-Keyboard Tutoring Sessions," *Proceedings of the 13th Annual Conference of the Cognitive Science Society, Chicago*, Hillsdale, NJ: Lawrence Erlbaum, pp. 576–580.

Seu, Jai Hyun and Martha Evens. 1991. "Understanding Ill-Formed Input to an Intelligent Tutoring System in an LFG Framework," *Proceedings of the Third Midwest Artificial Intelligence and Cognitive Science Society Conference, Southern Illinois University, Carbondale*, pp. 36–40.

Shah, Farhana. 1997. *Recognizing and Responding to Student Plans in an Intelligent Tutoring System: CIRCSIM-Tutor*, Ph.D. dissertation, Illinois Institute of Technology, Chicago.

Wiemer-Hastings, Peter, Arthur Graesser, Derek Harter and the Tutoring Research Group. 1998. "The Foundations and Architecture of AutoTutor." In Goettl et al., 1998, pp. 334–343.

Woo, Chong Woo. 1991. *Instructional Planning in an Intelligent Tutoring System: Combining Global Lesson Plans with Local Discourse Control*. Ph.D. dissertation, Illinois Institute of Technology, Chicago.

Yanofsky, Nancy M. 1978. "NP Utterances." In Farkas, D., W. Jacobsen and K. Todrys, *Papers from the Fourteenth Regional Meeting of the Chicago Linguistics Society, April 12–14, Chicago*, pp. 491–502.

Zhou, Yujian, Reva Freedman, Michael Glass, Joel A. Michael, Allen A. Rovick and Martha W. Evens. 1999. "What Should the Tutor Do When the Student Cannot Answer a Question?" *Proceedings of the 12th International Florida Artificial Intelligence Symposium (FLAIRS '99)*, Orlando, FL.