ARI Contractor Report 98-03

Dialogue-Based Language Training

Beth Plott
Micro Analysis and Design, Inc.

Dan Suthers and Beverly Woolf
Woolf and Company

Amy Weinberg and Bonnie Dorr
University of Maryland

March 1998

United States Army Research Institute for the Behavioral and Social Sciences

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1998	Final – Dec. 1, 1996 – June 1, 1997

4. TITLE AND SUBTITLE
Dialogue – Based Language Training

5. FUNDING NUMBERS
PE 0605502A
PA M770 TA2233

Contract No. DASW01-97-C-0012

6. AUTHOR(S)
Beth Plott (MAD), Dan Suthers, Beverly Woolf (Woolf and Company), Amy Weinberg, Bonnie Dorr (University of Maryland)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Micro Analysis and Design, Boulder Colorado. Woolf and Company, Amherst, MA; University of Maryland, College Park, MD

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Institute for the Behavioral and Social Sciences
ATTN: PERI-BR
5001 Eisenhower Ave.
Alexandria, VA 22333-5600

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
Contractor Report 98-03

11 SUPPLEMENTARY NOTES
This report is published to meet legal and contractual requirements and may not meet ARI's scientific or professional standards for publication.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*: The U.S. Army has recognized the need to transition the research on authorable tutoring systems and language learning, both funded by themselves and others, out of the research laboratory and into more applied settings. The objective of this project was to initiate a program of research to design a computerized tutor that is capable of teaching military personnel mission-relevant information and task performance through naturalistic dialogue between student and tutor.

The overall objective of this project was to demonstrate the feasibility of developing a general purpose, authorable, dialogue-based tutor. The central tasks were 1) the design of an artificially intelligent authorable dialogue system, 2) design of a lexical semantic authoring system, and 3) prototype development of an English natural language processing (NLP) system. Integration issues and data exchange methods for passing information between each of these central components were also designed.

14. SUBJECT TERMS
dialogue tutor, authoring system, knowledge base, NLP

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

Contractor Report 98-03

DIALOGUE-BASED LANGUAGE TRAINING

Beth Plott
Micro Analysis and Design, Inc.

Dan Suthers, Beverly Woolf
Woolf and Company

Amy Weinberg, Bonnie Dorr
University of Maryland

Advanced Training Methods Research Unit
Robert J. Seidel, Chief

United States Army Research Institute for the Behavioral and Social Sciences
5001 Eisenhower Avenue, Alexandria, VA 22333-5600

March 1998

Army Project Number
2Q665502M770

Small Business Innovative Research

Approved for public release; distribution is unlimited

FOREWORD

A major goal of computer-based training in general and language training in specific has been to simulate one-on-one dialogue with the computer acting as tutor. In recent years a small number of tutors that are capable of such dialogue have been developed as demonstrations using workstations. A recent breakthrough (which resulted from ARI's Military Language Tutor project) created a natural language processing engine and associated dialogue management system which can run in realtime on an affordable personal computer. This has made the development of a dialogue-based tutor possible for a personal computer. However, even on workstations, the lesson and associated dialogue content of such tutors require the major participation of computer scientists and computational linguists and this content typically is inextricably linked to the tutor software. The result is that a dialogue-based tutor requires very long periods of development and is more expensive than either the military training or general educational communities can afford. The purpose of the research described in this report is to provide a detailed conceptual design for a practical, fully authorable dialogue-based tutor. When implemented, it is likely to open a new line of research for the military and civilian training communities.

DIALOGUE-BASED LANGUAGE TRAINING

EXECUTIVE SUMMARY

Research Requirement:

A major component of U.S. Army readiness involves the skill and knowledge levels of the personnel who are responsible for operating and maintaining sophisticated systems, interacting with allies from other countries, as well performing intelligence operations. In order to skillfully perform their duties, Army personnel must initially be well trained and must also receive frequent opportunities to practice and refine their skills. Frequent training is required to limit degradation of skills, and maintain criterion levels of performance.

In order to maximize training effectiveness and efficiency, training exercises should be structured to provide each trainee with maximum learning gain from each training session. The goal of each training session is to build upon the existing knowledge of the trainee, to diagnose and correct deficiencies as efficiently as possible, and to allow for consolidation of skills through practice. The training effectiveness of these sessions would be increased with the implementation of instructional strategies and technologies that promote efficient acquisition and retention of skills and knowledge.

The U.S. Army has recognized the need to transition the research on authorable tutoring systems and language learning, both funded by themselves and others, out of the research laboratory and into more applied settings. The objective of this research was to initiate a program of research to design a computerized tutor that is capable of teaching military personnel mission relevant information and task performance through naturalistic dialogue between student and tutor.

Procedure:

The overall objective of this project was to demonstrate the feasibility of developing a general purpose, authorable, dialogue-based tutor. The central tasks were 1) the design of an artificially intelligent authorable dialogue system, 2) design of a lexical semantic authoring system, and 3) prototype development of an English natural language processing (NLP) system. Integration issues and data exchange methods for passing information between each of these central components were also designed.

Findings:

The research effort determined that it is feasible to link a English NLP system to a AI dialogue system to develop a robust dialogue-based tutoring system. The effort succeeded in not only determining that development of a authorable dialogue-based tutor is feasible, but that the NLP would enhance the capabilities and usability of a AI dialogue system through the expansion of acceptable student responses.

Utilization of Findings:

The research provides a framework for development of a tutoring system that could be used to teach mission relevant information though the use of naturalistic dialogue. The information to be taught would be authorable by subject matter experts with no computer programming or computational linguistics background.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Overview

This report documents the specifications to develop an authorable dialogue-based military information and procedural tutor. Specifically, the focus is on the concept for development of an authorable dialogue system capable of supporting military and procedural informational needs. In support of the conceptual dialogue system, a English natural language processing (NLP) system would have to be developed and integrated with a artificially intelligent knowledge base. The concept is based upon the successful design and proof of concept analysis that was performed in a Phase I Small Business Innovative Research (SBIR) effort. That effort determined that it is feasible to link a English NLP system to a AI dialogue system. The Phase I effort succeeded in not only determining that a development effort is feasible, but that the NLP would enhance the capabilities of a dialogue system through the expansion of acceptable student responses. The specification also details a design that would allow nonlinguists to add nouns to the lexical conceptual structure (LCS) portion of the conceptual system.

Results of the Phase I Work

In the Phase I SBIR, the overall objective was to demonstrate the feasibility of developing a general purpose, authorable, dialogue-based tutor. This work was done to "set the stage" for a potential development phase, whose objective would be to construct and test a fully functional system. The Phase I effort resulted in the successful completion of six main tasks.

- Task 1: Design a Prototype Dialogue System
- Task 2: Develop NLP tools to Support Dialogues
- Task 3: Develop Design for LCS Editor
- Task 4: Resolve Integration Issues between Components
- Task 5: Develop Screen Prototypes
- Task 6: Investigate Internet Delivery

Each of these tasks are discussed in this section.

Task 1: Design a Prototype Dialogue System

One goal of this task was to design a English NLP system that preserved the present capacities of the previously developed MILT tutor and also enhanced its capabilities. The main augmentations would be the introduction of a dialogue component capable of supporting meaningful lessons to teach skills of relevance to the military, and the capacity to make this dialogue system and its associated instructional software as authorable as possible without compromising the instructional effectiveness of the system. These were the two chief goals that motivated the design of the system. This process involved first choosing a prototype domain that would involve generally useful and authorable components.

The main educational strategies and facilities of the dialogue section are as follows. Students would practice emergency medical decision making skills using a simulation of an unconscious patient in the battlefield. Natural language help would be available on demand during practice. Subsequently, students would reflect on their performance while interacting with a natural language based tutor that would help the student assess his or her performance. This post-performance reflection dialogue would follow a typical, authorable format but would be modified to suit the context (e.g., the student's performance). The separation of practice and reflection is intended to increase the realism of the practice, and to maximize cognitive resources available for learning during reflection.

The dialogue capabilities would be augmented by the English NLP system. In Phase I the English NLP system was designed to be compatible with the dialogue architecture. This means that the NLP would allow dialogue representation requirements to be met while allowing students to provide synonymous answers and paraphrases to questions. The English NLP designed would take advantage of the LCS system previously developed for the MILT project.

Task 2: Developing NLP tools to Support Dialogues

The University of Maryland has developed a version of the REAP (Right Edge Adjunction) parser used in the MILT as the parsing backbone of this system. The lexicon currently used for English contains approximately 50,000 head words derived from Webster's dictionary. Part-of-speech is encoded and used to derive morphologically inflected forms, for instance past tense and plural. Appropriate spelling rules for English are encoded using Alpnet, a two-level Kimmo-like system, only faster and more powerful. Fast parsing is crucial to this effort since the tutor must contain an extensive knowledge base, semantic and discourse components, as well as the syntax and morphology. In Phase I. a English NLP system was chosen and the algorithms and tools required to expand the coverage of the REAP system were designed. During development, the coverage of the system would be expanded so that it could be used with the dialogue system in a large variety of application domains. Linking routines would also be written between the NLP and dialogue modules for both analysis and generation. Further details are given in Task 2 of the concept specification description.

Task 3: Design a lexical conceptual structure (LCS) authoring system

The design of a system that would allow authors to add new nouns to the lexical conceptual structure (LCS) was completed in Phase I. During Phase I the algorithm to search the WordNet database and extract synsets was completed and implemented. During development of the conceptual system (Task 2), the relevant editors would need to be built and the system would require porting to the MILT.

Task 4: Resolve integration issues between components

All components designed including the knowledge base, English NLP, and LCS Authoring system would need to be integrated. During Phase I, the algorithms and data exchange protocols that would be necessary for the various components to function as a complete system were determined.

Task 5: Develop screen prototypes of interfaces to be built

Examples of the screen interfaces for the LCS authoring system and for the dialogue authoring system are shown later in the report.

Task 6: Investigate internet delivery

An objective of this design was for the developed dialogue-based tutoring system to be accessible either as a stand alone system or through distributed, Internet environments. This task involved the investigation of various methods to distribute the tutor and/or lesson files via the Internet. It was discovered that the most straightforward method would be to establish a world wide web page for the tutor. During Phase I a sample proof-of-concept page was constructed. The page is located at http://www.maad.com/MILT. For this task, a HTML layout page that allows users to download the users manual, report suspected bugs, provide suggestions, and obtain new information was developed. It is envisioned that the site could be expanded and used as a depository for lesson materials, multimedia resources, knowledge base definitions, and LCS noun entries. Any technical support questions would also be sent to the address.

Technical Objectives and Conceptual Approach to Developing a Dialogue Based Tutoring System

The remaining sections of this report document a conceptual approach to developing a dialogue based tutoring system that uses NLP and AI technology. The approach is based upon concepts proven feasible in the Phase I SBIR effort. The objectives required for the development of the conceptual system are:

- Objective 1: Develop a fully functional English NLP system
- Objective 2: Develop authorable dialogue lesson capability
- Objective 3: Develop LCS authoring tool

- Objective 4: Integrate all developed pieces into a functional tool
- Objective 5: Develop commercial interest in the developed dialogue-based tutoring tool

Conceptual Development Work Plan

The work plan is designed to discuss the work that would be required to expand and incorporate the innovative technologies developed during the Phase I SBIR effort into a tangible product that would provide a value added capability for training developers. The work accomplished during the Phase I SBIR answered many of the concerns about the capability of providing this innovative technology and have reduced much of the risk that would be inherent in such an effort.

The following tasks would be required to develop an authorable, dialogue-based information and procedural tutor:

- develop an artificial intelligence system that would allow instructional personnel to easily develop, expand and maintain knowledge representations
- develop a system that would allow users to add new nouns to the lexical semantics structure
- develop a full English language Natural Language Processing system

The innovations could be integrated with an existing and easy to use authorable tutoring system (MILT). The integrated tutor would allow instructors to easily develop lessons in a wide variety of domains. The completed conceptual tutoring system would include: 1) an authorable knowledge base that would allow the addition and modification of domains, 2) a state of the art English natural language processing system, 3) an authoring interface to the lexical semantics structure that would allow non-linguists to enter new nouns, 4) a dialogue management and language generation system, and 5) a general-purpose instructional authoring module that would allow instructional developers to develop training material in a wide variety of domains. This tool would significantly advance the state-of-the-art in dialogue-based authoring tools and tutors.

The development of the conceptual system would be accomplished through the performance of the following tasks:

Task 1. Determine Target Application Domain
Task 2. Develop a robust English NLP system
Task 3. Develop a LCS Authoring System
Task 4. Develop a Dialogue System
Task 5. Integrate Components
Task 6. Expand Internet Capabilities
Task 7. Generate Military Based Dialogue Lesson

There are three major efforts that would be required to accomplish the objectives. The first would be to develop the natural language processor capabilities required to support a robust dialogue system (Tasks 1, 2, 3, and 5). The second major effort would be to develop a dialogue tutoring system that could be authored for similar domains (Tasks 1, 4, and 5). The third thrust would be to generate commercial interest and determine the commercial requirements that would expand this technology into a viable product (Tasks 1, 6, and 7).

Task 1: Target Application Domain

One of the first steps required before the conceptual dialogue system could be designed was to choose a target domain and the educational problem to be addressed through natural language interaction.

Educational Problem

During Phase I, emergency medical treatment was chosen as a target domain. The Army has major training problems in the medical arena. For example, training a student to the level of a licensed practical nurse or nurse requires years of study at great expense. Most army bases send candidates to community colleges or nursing or medical schools. Often 50% of the students fail out of the program or transfer to other bases. Personnel at Forts Leavenworth, Devens, and Sam Houston are looking into distance learning technology and adaptive technology to support learning on base. Currently the 91 Charley MOS program, similar to an LPN program, contains courses in anatomy & physiology, pharmacology, nutrition, physical therapy, surgery prep and various therapies. One goal is to move this 91 Charlie course to the individual home sites through computer training. Col. Frank Coppola, Fort Leavenworth, Kansas, who is on the Army Science Board, Bob Clayton, Office of the Command Surgeon, U.S. Army, Ft. Bragg, and Col. Joyce Humphry, emergency medicine specialist, at Ft. Devens, MA are working to include more computer based training in the Army's medical programs.

In addition to the Army training need, the emergency medical domain was chosen for the following reasons:

- The domain is sufficiently constrained to limit the knowledge and language engineering needed for a prototype demonstration.
- The team had prior experience with the domain.
- Different kinds of dialogues are possible: the domain could support demonstration of various interaction styles.
- The domain has commercial application

The conceptual system described would address the problem of teaching nonspecialists (e.g., infantry) basic emergency medical procedures, so that they may better help injured comrades on the battlefield and in other emergencies. Students would learn the basics of "ABC-BS" - Airway, Breathing, Circulation, Bleeding, Shock. They would learn this sequence of priorities; how to check for these conditions (e.g., blocked airway, lack of breathing, lack of pulse, signs of shock) and how to treat them using minimal first aid equipment. Hands-on practice would not be emphasized in this system. Rather, students would be asked to make decisions under pressure, and subsequently review their performance. Interaction with the system would fall into two phases: practice and reflection. During practice, the student would practice emergency medical decision making skills using a simulation. During reflection, the student would interact with a natural language based tutor to assess the student's performance. This educational problem has parallels in other military decision-making under pressure situations, especially those where there is an established procedure to follow. For example, Woolf and Company has built a system for teaching Advanced Cardiac Life Support (ACLS) skills using prescribed protocols. It is expected that the work would generalize to similar kinds of training situations, and furthermore, it is expected that the dialogue techniques would generalize to other domains beyond these kinds of situations.

Design of the Interactive Experience

Once the domain was decided upon, two possible scenarios were considered: dialogue with a conscious patient, and dialogue with a tutor while treating an unconscious patient. Dialogue with a conscious patient had the advantage that the skill learned is a dialogue skill, so NLP would not be secondary to the task -- the system would not be subject to criticism that NLP is not the right way to teach. However, such a dialogue is less constrained and thus more difficult to design and implement an authorable system. Also, it does not use NLP to teach directly, so could not demonstrate pedagogical dialogue strategies. Dialogue with a tutor while treating an unconscious patient presented several somewhat independent possibilities: the student could ask the tutor questions; the tutor could monitor student during performance for suboptimal behavior and intervene; or the tutor could engage the student

in a reflective follow-up analysis of student performance. These options are discussed later. The advantages of dialogue with a tutor are that teaching is done *with* NLP, so it could demonstrate pedagogical strategy in dialogues. A possible disadvantage is that the task to be learned is not an NLP task. Instead, NLP would be used for tutoring. Hence the system would be subject to criticisms that NLP is not the best way to teach the primary task (of emergency medical treatment).

Practice of emergency medical skills.

In the described system, the student would be presented with a situation in which an injured patient (soldier) is found unconscious on the ground in a combat situation. The student would need to quickly assess the patient's status and apply appropriate life-preserving treatments. This would be done under time pressure and with simulated distractions. Beginning students would be provided with a simulated "mentor" medical technician who could be consulted with questions. The practice session might end with successful dispatch of the patient in an emergency transport vehicle or hand stretcher, or in the death of the patient.

Reflective follow-up.

The student would be in a simulated debriefing situation in which a mentor medical technician or paramedic reviews the student's performance using natural language dialogue interaction as well as replaying "video clips" of the practice session (the simulation is rerun in an insert window). The dialogue would be controlled primarily by the mentor, who would ask the students questions concerning his or her performance. The student would be asked to self-assess performance using a set of assessment criteria. The mentor could acknowledges accurate self-assessments (whether positive or negative) where appropriate, and might provide elaboration if the student's assessment was incomplete. The mentor could also point out problems that the student may not have recognized, and describe recommended procedures and their justifications. Visual aids, e.g. diagrams or graphs, could be displayed as needed to support the mentor's advice.

Simulation of Emergency Medical Situation.

The described NLP tutor system leverages off of prior work done in the arena of medical diagnosis and treatment. Woolf and Co. has built a training system based on a real-time simulation in the area of Cardiac Arrest (Figure 1) which integrates a numerical simulation and knowledge-based modules, including a user model, planner and plan recognizer (Eliot & Woolf, 1996a; Eliot & Woolf, 1996b). The system reasons about a user's knowledge and skills making assumptions about the user's state of knowledge while he/she works within the simulation, trying to save the life of a "patient" undergoing a cardiac arrest. Based on the student's strengths and weaknesses in treating the "patient," the curriculum would be biased towards the desirable learning states.

The tutor reasons about medical constraints of cardiac arrest and forms goals to teach specific topics to individual students. Each topic is taught only when the real-time simulation reaches the proper state, in terms of clinically appearing cardiac rhythms, and new simulations generated based on observed learning needs.

The Tutor includes an accurate descriptive model of the emergency room environment and general patient status, combined with a causal model of cardiac function and related physiologic systems. The user model uses both a complex representation and a sophisticated control structure to supply responsive and flexible but canned comments. The goal was not only generation of immediate responses in a single session, but also the improvement of instruction in the long term between sessions.

The tutor is based on the following four steps:

1. goal selection--reason about the user's needs and the best context for meeting them. A goal includes new topics and new problem cases.
2. plan formation--move from one scenario or problem case to another selected to best satisfy a goal

3. plan instantiation--simulate a new context or provide a new "patient" with dynamically generated medical problems to satisfy the goal
4. simulated reasoning --apply situation-specific knowledge within the simulation to help achieve the user's goal

Each "patient" presents a unique set of medical problems and history in order to ensure that the user is learning new material. The tutor was evaluated with 50 medical students with very positive results.

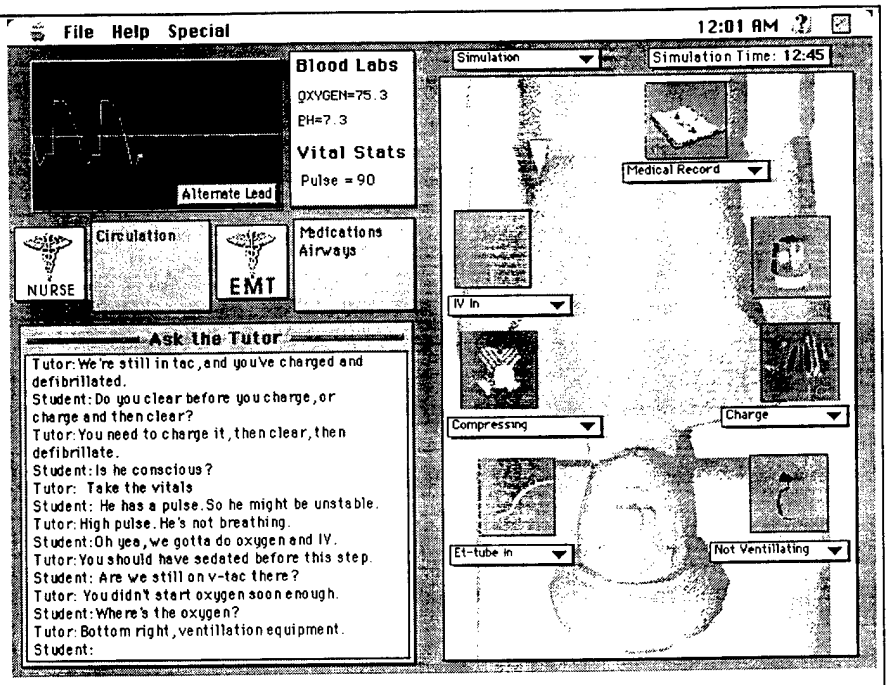


Figure 1. The cardiac tutor

Dialogue Types Supported

The student would be engaged in NL dialogue while working with the tutor. He/she would be able to ask the tutor context-dependent questions, including questions about patient care, etc. The natural language dialogue in the system would take place through a keyboard and textual display area. Subsequent implementations could use speech recognition and speech synthesis for spoken dialogue.

Below, different dialogue types that could be modeled are categorized. Subsequently, the choice of which to model for the conceptual system is discussed.

Question Answering. Questions could include context-independent questions concerning factual knowledge (e.g., anatomy and physiology), and context-dependent questions concerning the patient's current state and proper procedure. Simple question answering would be straightforward to design and implement, but dialogue is limited.

Coaching. The tutor would monitor student performance during a session, and interrupt as it deems necessary to address performance errors or to otherwise take advantage of learning opportunities in real time. While appropriate for coaching some types of skills, this may not be appropriate for demanding real-time performance skills.

Critiquing. The tutor would monitor student performance quietly during a session. At the end of a session, tutor would generate a monological critique of student's performance. Monologues in themselves may not be interesting enough.

Reflective Follow-up. The tutor would conduct a post-performance dialogue with student interaction. The tutor would have primary control of the dialogue, commenting on performance and asking students particular questions, but students would have the opportunity to redirect the dialogue. This form of dialogue is appropriate but more ambitious.

Reflective follow-up has been shown to be an effective aid to learning (Sherlock, Katz and Lesgold 1993). The cognitive load generated by rapid decision making during emergency treatment is too high to expect students to also engage in reflective learning during performance. NL interruptions could be seen as disruptive (unless NL is the skill being learned, as is the case when simulating conscious patients and/or other emergency personnel rather than a tutor). The student could allocate more cognitive resources into assessing performance and learning if this is attempted after the performance rather than during. For these reasons, modeling reflective follow-up dialogues, controlled by the tutor but allowing questions as interruptions, was chosen for this design. Another theory - model tracing - suggests an alternate approach. Model tracing uses a detailed model of expert behavior, and traces student performance with respect to this model. Errors are flagged immediately when they occur, under the rationale that learning occurs in context and errors don't get out of hand, frustrating the student. This approach has been applied only to domains without real time performance requirements (e.g., Lisp programming and geometry proofs). Its appropriateness for real-time domains such as emergency medical treatment is less clear. Although computer simulations enable learners to pause real time performance to engage in learning, the tradeoff is less realism in the real-time demands of the task.

Development of the described system would support question answering on demand during practice, and support tutor-guided reflective follow-up with interruptions for clarification and questions and background knowledge allowed.

Authorability of System by Non Specialists

Authorability of the tools developed for this conceptual system is required because the goal is to allow non computational linguists or computer scientists to reconfigure the system for use in domains beyond that of the initial system to the greatest extent feasible. Two research approaches could be used to develop authoring tools for the NLP medical tutor described above. These tools would enable instructors and programmers to create new content for building new tutors and would increase production of natural language training systems.

Authoring tools for adaptive instruction.

Tools would be developed for authoring separate parts of the training systems, including the domain and curriculum knowledge base, the NLP front end, the student model and the interface. Both generic shells and authoring tools would be required. Shells are the remains of existing tutoring systems from which the content has been removed and the algorithms and data structures remain. The underlying architecture is kept to be reused with new content material. Shells make the programming task more efficient, but are not intended for instructors; they are used primarily by programmers. Authoring tools allow non-programmers--especially instructors, to develop new systems. They are more difficult to build than shells, but have a higher payback. A number of authoring tools and shells could be built, some addressing very general needs such as student record keeping, and others facilitating rapid production of tutors in specific areas. Combining authoring tools and shells, nonprogrammers would be able to author lessons for tutoring systems, providing greater pedagogical flexibility than is usually associated with authoring tools.

Reusable instructional components and tools.

A reusable and interoperable design would enable instructors to combine existing functional modules when constructing new instructional systems, and to utilize commercial software where applicable. Possible components include knowledge bases of questions and answers, a medical lexicon, quiz templates and active diagrams with plug and play interoperability. Reusable components fill a

critical need as natural language training systems scale up. The goal of the conceptual system is to eliminate the need for instructors to have to "reinvent the wheel" with each new system. For example, basic anatomy terms would carry over from one medical field to another. Components would improve the effectiveness and reusability of future NLP systems, allowing instructional designers and students to use existing components that contribute to their own classes. The specification draws upon experience constructing interoperable systems, as well as on recent standards efforts for computer-based education. Resource retrieval facilities in our authoring tools would enable instructors to quickly answer questions such as: "Does the thing I want to create already exist?" and "How can I use what is already there?" Authoring will be discussed in further detail in the following sections.

Task 2: Develop English NLP

All of the dialogue activities discussed in Tasks 1 and 4 would require supporting NLP components. Below an LCS based synonym finder is discussed (Task 3) that would clearly be important in allowing the tutor to score student answers correctly, and to understand when the student is in control of the concepts (s)he needs to learn. In addition, a broad coverage morphological analyzer and parser would be required to analyze student input and to serve as the interface with lexical- conceptual and higher level semantic representations.

Examples of English morphology.

As mentioned earlier, the current system contains ~50K words, which spans a significant range. Shown below are the derivations for two extreme words 'aardvark' and 'zymosis'.

```
********* Solution 1 **********
Lexical: #aardvark+#              Exploded: '#aardvark+#'
Surface: aardvark                 Key: aardvark
Translation: [aardvark][+]
Features:  deriv no , cat n , type common , num sing
key: aardvark              gloss: aardvark
1 unique keys
1 solutions
elapsed time = 0 seconds

********* Solution 1 **********
Lexical: #zymosis+#               Exploded: '#zymosis+#'
Surface: zymosis                  Key: zymosis
Translation: [zymosis][+]
Features:  deriv no , cat n , type common , num sing
key: zymosis               gloss: zymosis
1 unique keys
1 solutions
elapsed time = 0 seconds
```

Explanation of fields in the derivations:
"Lexical" this is the internal representation of the word
"Surface" is the external, 'spelled', form of the word
"Exploded" is the same as "Lexical" for English
"Key" is the same as the 'root' of the word
"Translation" shows the gloss for each morpheme of the word
"Features" shows the syntactic features for the word
"key" is the root of the word, generally the first morpheme
"gloss" is generally the first gloss.

Within the "Features" list, there are numerous feature value pairs. The meanings for these are:

"deriv no", this word is not derived. An example of a derived word might be "aardvarkable", derived in 2 steps: 1) aardvark the verb is derived from the noun via the null-morpheme, and 2) aardvark the verb takes the '-able' ending and becomes an adjective. The ability of the current system to derive words is significant, but currently only words with "deriv no" are used by the parser.

"cat n", this word is a noun, e.g. category noun.

"type common", this word is a common noun.

"num sing", this is the singular, as opposed to plural, form of the word.

The results from morphology are encoded in a Lisp-like notation for the parser. The actual entry for 'zymosis' is presented below.

```
(zymosis (syn ((root zymosis) (gloss zymosis)  (deriv no) (cat n) (type common)
(num sing))))
```

Irregular Verbs

The past tense, and past participal for over 100 irregular verbs are included. Notice for the examples below, that the "key" is the infinitive form of the verb. Words that have been looked up in the lexicon and morphologically analyzed are the input to the parser.

Lexical	Exploded	Surface	Key
#wrote#	'#wrote#'	wrote	write
#is#	'#is#'	is	be
#went#	'#went#'	went	go

Table 1. Irregular verb examples

REAP is an extremely efficient parser for natural human languages. This parser is based on a model of the human language processor (HLP) that allows it to be both fast and cover a variety of actual languages. For instance, systems have been developed for Arabic, Spanish, and English using this parser, and a Korean system is under construction. This range of languages presents a number of problems for any theory of parsing, but the current parser handles these problems with ease, and achieves parsing speeds in excess of 100 words per second. These results are the consequence of designing a parsing algorithm that mimics the HLP, and thereby exploits the same characteristics of the input signal that allow the HLP to parse in linear time.

Characteristics of a Deterministic Parser and Relevance to the Efficiency of the System

In a deterministic parser, all structure is permanent. That is, once a structural decision has been made by the parser, it cannot be rescinded. This characteristic of deterministic parsers contributes to their efficiency (Berwick and Weinberg, 1985). This efficiency derives from the fact that determinism limits the number of analyses the parser pursues.

The high degree of ambiguity in natural language appears on the surface to make determinism impossible. For example, the modification of structure in (1) should not cause problems for the language processor even though the structure is ambiguous until after the third word.

The two structures that are possible are given in (2).

```
1a) I knew Eric.
1b) I knew Eric was a great guy.

2a) [vp believe Eric]
2b) [vp believe [S Eric was a great guy]]
```

Early theories of parsing assumed that the parser used phrase structure rules like those in (3). Building the correct structure under this assumption entailed choosing the correct phrase structure rules. If rule (3a) was chosen, sentence (1a) would be accepted, but (1b) would not.

3a) VP -> V NP
3b) VP -> V S

As the interpretation of phrase structure rules changed (Stowell 1981), new processing strategies were pursued. D-theory (Marcus, Hindle, and Fleck, 1983 [MHF]), captured a wide range of troublesome data with a single insight: by replacing 'directly-dominates' with 'dominates', additional intervening structure could be added during the course of a parse without rescinding previous decisions. For instance, consider how the distinction between (2a) and (2b) is captured by D-theory. 'Eric' is dominated by the VP of 'knew' in both (2a) and (2b) as seen in (4). Therefore, the dominance relation between 'Eric' and 'knew' does not change.

4a) I knew Eric.
4b) I knew Eric was a great guy.
4c) I [vp knew [np Eric]]
4d) I [vp knew [vp [np Eric] [v' was a great guy]]]

Construction of an Efficient Deterministic Parser

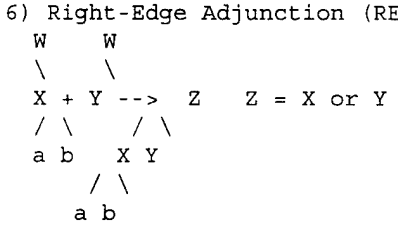
In this section a Right-Edge Adjunction Parser [REAP] is discussed, which can parse a sentence and build a syntax tree deterministically. A REAP is a special case of an incremental tree building parser, that allows adjunction to the right-edge of the current tree as its only syntax building operation. This parsing model borrows heavily from the work of MHF (1983), Weinberg (1993a), and Gorrell (1994a). While this processing model is built on D-theory, several additions and modifications are offered, which improve its coverage of difficult sentences and allow for an efficient implementation.

As the name implies, the principle structure building operation of a Right-Edge Adjunction Parser (REAP) is Right-Edge Adjunction (REA). Right-Edge Adjunction is adjunction of a input constituent to a right-edge node. Right-edge nodes are those nodes along the right-edge of the tree (see 5)).

5) à is a Right-Edge Node if,
  a) à is the rightmost child of a right-edge node, or
  b) à is the root node.

Nodes can be added to the tree only by adjunction to a node on the right edge (as shown in 6)).

6) Right-Edge Adjunction (REA):



In addition, the criterion of Linearity is imposed. Any attachment site not available by Right-Edge Adjunction would violate at least one of: linearity, binary branching, or determinism.

Linearity.

The linear order of lexical items in the input string must be identical to the linear order of the lexical items in the corresponding tree.

The definition of Right-Edge Adjunction naturally lends itself to a system that allows only binary branching. As mentioned earlier, binary branching would be used, and no non-branching nodes would be allowed.

Parsing algorithm.

As each lexical constituent is encountered:

a) the item is placed in the tree by the syntactic processor at the "best" site using Right-Edge Adjunction, and

b) isomorphic operations are performed by the constraint processor on the resulting tree.

The "best" site stated in the parsing algorithm above represents the structural determinism of the proposed model. While multiple sites might be available, only the "best" is chosen. If that happens to be the wrong attachment site, a garden path sentence will result. To determine which site is the "best", the Priority scheme presented below is used.

```
Priority of Attachment Sites (1 = highest priority):
  1) Priority of grammar rule
  2) Theta-assignment
  3) Recency (and Frequency)
```

A simple example is presented next that demonstrates the concepts in this section:

```
7a) Mary sees John.
7b) [np Mary]
7c) [vp [np Mary] sees]
7d) [vp [np Mary] [v' sees [np John]]]
```

When 'Mary' is encountered, there is no tree yet, so it must be attached at the root. When 'sees' is encountered, there is only 1 possible attachment site, as sister to 'Mary'. However the relationship between 'Mary' and 'sees' is an open issue. After the syntactic processor has attached 'sees', the constraint processor determines the relationship between 'Mary' and 'sees'. In this case, the relationship is subject-verb. When 'John' is encountered, there are 2 possible attachment points: 1) the lower position as sister to 'sees' or 2) the higher position as sister to 'VP'. In the lower position, 'John' can be taken as the object of 'sees', while in the higher position there is no reasonable interpretation. Therefore, the lower position is chosen by the syntactic processor. The constraint processor then assigns the verb-object relationship between 'sees' and 'John'. With the resulting tree as seen in (7d).

Implementation Possibilities

Building structure and performing licensing operations could have consequences even for previously built structure. The following paragraphs describe the types of consequences that are allowed. As stated earlier, the shape of the tree is fixed except for the addition of new material added to right-edge. This is consistent with Gorrell's (Gorrell 1994a) structural determinism. However, node labeling could change. Node labeling consists primarily of category and level, as well as other linguistic features (number, case, etc.). Node labeling is the result of projecting the node's head. A node's head is either its left or right child, following the projection principle. There is no reason for the theory to independently restrict any change in node labeling, including preventing the head from changing. That is, any change in node labeling is allowed as long as the change is the consequence of some allowed operation. For instance, as structure is added and licensing is performed, higher nodes are re-examined. As a node is re-examined, nothing in the theory prevents the head for this node from changing. While this in not likely to happen, any restriction is unnecessary and artificial.

The category of a node is, generally, the category of its head. Therefore, the category of a node could change under two conditions, 1) it changes its head as described above, or 2) the category of its head changes. Typically, the category of a node would change as possible senses of its head are removed by disambiguation. A sense for a word is removed from consideration if there is no rule which allows that sense in the current configuration. Once a word sense is removed, it cannot be re-accessed. This is consistent with psycholinguistic evidence (Carlson and Tanenhaus, 1988). An example of word sense disambiguation is given below. If 'fire' with both the noun and verb senses were placed in the tree with 'the' as its sister, no rule would allow the verb sense in that configuration. Consequently, the verb sense would be eliminated.

```
8a) [the n-fire] -- allowed
8b) [the v-fire] -- no rule allows verb here
```

Finally, node labeling information could be structural in nature. That is, linguistic information could be the result of structure rather than inherent to the head. For instance, an important structural feature is X-bar level. The structural information for a node is determined by rules, and could change if the rules which built a node change. This would generally happen when the disambiguation of a node's child causes a rule which formerly applied to no longer apply.

Complete vs Incomplete Rules

The rules that are used to build structure follow from licensing configurations. However, not all rules represent actual licensing configurations, some represent potential licensing configurations. Consequently, rules are considered "complete" or "incomplete".

"Complete" rules represent satisfied licensing configurations, while "incomplete" rules represent potential licensing configurations. "Incomplete" rules allow attachment of material which precedes the construction of a correctly licensed projection. "Incomplete" rules can only refer to leading edges of constructions that occur in "complete" rules. There are two kinds of "incomplete" rules (for English): 1) rules which refer to pre-head material, and 2) rules which refer to unprojected heads.

For instance, if a "complete" rule allowed an NP to be the right sister of a V0 (the notation [V0,NP] is used to represent this pattern), then this would imply that there should be "incomplete" rules for all pre-head material for an NP. Consequently, there would be an "incomplete" rule [V0,D], where D= determiner. This would follow from the observation that D is a valid leading edge of NP, as in the derivation [D + N] -> NP.

An example of the second type of "incomplete" rule would be [V0,PP] -> [V0,P0]. That is, if there were a "complete" rule that allowed a PP as the right sister of a V0, then there would be an "incomplete" rule the allowed a P0 as the right sister of V0. Again, this follows from the straight-forward observation that P is a valid leading edge of PP, as in [P + NP] -> PP.

Nodes are also either "complete" or "incomplete". Nodes built using "incomplete" rules are themselves "incomplete".

```
9) A node is "complete" if
   - it is a lexical node, or
   - it was formed by a "complete" rule and both of its children are "complete"
```

For instance, the node formed by the rule [V0,D], given above, would not be complete, since the rule that formed it is not complete. In other words, more material needs to be added below this node before it is a proper licensing configuration. Consequently, no material can be attached to or above an "incomplete" node.

The parser was used to handle a sample corpus relevant to this application with great success. In the Appendix to this report, sample output from the parser and LCS system are presented.

The LCS Component:

The University of Maryland parser feeds into a Lexical Conceptual Structure (LCS) component that handles the lexical semantics of the system. The LCS indicates the meanings of words in a compositional way. Words that share features of meaning would share parts of the associated LCSs. The advantage of the LCS matching becomes clear in the following example. It would be a tedious task if the author were expected to specify, in advance, all possible ways that the student might choose to convey the desired information.

Another example illustrating this provided by an example from the preliminary corpus that was analyzed during PHASE I. A student might be asked for a description of a patient's condition and reply as follows:

```
("That does not look like vtac. " 'ENGLISH)
```

This would be given the following LCS:

```
((:ROOT STATE BE PERCEPTUAL NIL 1
  ((:SUB THING REFERENT NIL NIL 2) (:ARG POSITION AT PERCEPTUAL NIL 3
((:SUB THING REFERENT NIL NIL 2) (:ARG THING VTAC NIL NIL 4)))
```

```
    (:MOD MANNER LOOKINGLY NIL NIL 5)))
  ((4 (GLOSS "like") (SURFACE "like") (ASPECT IDENTITY)
    (CASE NOM
     ACC
     DAT
     GEN
     REFL)
    (PERSON THIRD) (GEN F M) (NUM SING) (HUMAN -) (GLOSS "vtac")
(SURFACE "vtac"))
```

It is desired that the dialogue manager accept answers like "It seems like, appears to be, 'looks like' vtac. This could be easily specified in this compositional notation by calling for a "State: Be perceptual" type verb for the discourse operator.

Thus the LCS component gives a representation of the literal meaning of sentences to be used in the dialogue lesson. This would be the input to the dialogue manager, which would use these meaning representations to help decide what the student has said , and to structure its own response based on this information and its own discourse planning tools and student model. This representation is appropriate because it allows maximum flexibility in specifying what counts as a synonymous or appropriate response. The section below shows that a different mechanism to handle synonymy of non predicates (nouns) would be provided.

Task 3: Develop a LCS Based Synonym Finder

The goal of this task would be to enhance the authorability and effectiveness of the Matcher that forms part of the MILT system. The Matcher supports question answering exercises that are embedded in the MILT foreign language tutor. The goal from the beginning of the development of MILT was to make design of these exercises easier for instructors. This is why storing of semantic representations for answers as opposed to storing the answers as strings was chosen. The lexical semantics of the current version of the MILT is based on Jackendoff's theory of Lexical Conceptual Structures (LCS) (Jackendoff, 1983; Jackendoff, 1990) as enhanced and implemented by Dorr, (1993); Dorr et al. (1995). A tutor with LCS representations is superior to other string based or non-semantically enhanced tutors in its support of question answering and dialogue exercises. In the ideal case, it is desirable that an instructor be able to store the expected answer to a question in a question answering exercise, and allow the student's answer to count as a match to this answer whether or not it is a word for word match, or is synonymous to the instructor's stored answer. The present system meets this requirement for predicates (verbs and nouns based on verbs) and so can deal with situations like the following:

```
We are preparing to attack X
We are preparing an attack on X
```

The system deals with synonymy relations by having the Matcher (the component that decides whether a student's answer matches an instructor's stored expression) match LCSs, rather than strings. In the case above, a match is declared if there is identity at a certain level in the tree structure rather than total string identity.

So, for example, the LCS corresponding to the portion of the sentence that includes the verb phrase "attack X" will match the LCS corresponding to the portion of the sentence that includes the noun phrase "attack on X":

```
[Event GO_Loc
([Thing 1],
 [Path TOWARD_Loc ([Position CO_Loc ([Thing 1], [Thing 2])])],
 [Manner ATTACKINGLY])]
```

Our goal for this part of the initiative is to add the capability to automatically add nouns and synonyms for the purposes of matching. . For example, a student being asked :

```
(10) Where did you spot the soldiers?
```

might answer

```
(10a) in the forest or
```

(10b) in the woods

It is desirable for the tutor to be able to score both answers as correct even if the instructor has only listed (10a) as an appropriate response.

In order to have this capability, instructors must be able to add new nouns to the lexicon and have them correctly decomposed morphologically and associated with semantically related words. This editor must be maximally simple and must not require deep knowledge of either LCS representations or other aspects of computational linguistics. Four subtasks are involved in accomplishing this goal. The first subtask was concentrated on during Phase I. the team has experience with the second subtask (reported in Dorr et al (1995)). The other two subtasks have been designed but would have to be accomplished. These subtasks are:

1. Design a mechanism for navigating a large scale database of synonyms in a way that would make the synonym class conveniently available for use during matching. This database is in English.
2. Provide a utility by which authors could add synonym sets to the class of known nouns.
3. Translate the root forms of nouns into the target language using the large scale dictionaries available for Arabic or Spanish.
4. Provide appropriate morphological information so that the word could be recognized by the parser.

During Phase I, a mechanism was devised that would automatically extract synonyms for words in the matcher exercise. A method was devised whereby the tutor would accept all nouns in the synset (synonym set) of the correct nouns in their respective sentence placement. For example, "put the trash in the trash can" would be correct if the master sentence was "put the trash in the garbage can", since trash can and garbage can are synonyms. A database of nouns and their senses originally created by George Miller and his colleagues at Princeton for Wordnet 1.5 was used. This database was modified to remove extraneous information for the purposes of our research. This resulted in two lists being created:

1. a list containing nouns followed by senses of meaning appropriate to each noun. The senses are indicated by 'sense reference numbers'. For example:

```
acceleration 3524146 185892 9175297
accelerator 2041426 2040820 2040999
accelerator_pedal 2041426
accelerometer 2041664
accent 4611417 8721456 4582727 4623634 4447170
```

and 2. a list of senses of meaning followed by the nouns that belong to each one:

```
00185892  acceleration quickening speedup
00186037  pickup getaway
00186118  scud scudding
00186230  translation displacement
00186330  transplant transplanting
00186449  shift shifting
```

Note that acceleration in the first list has several numbers (the sense reference numbers) following it. The second number is 185892.

Now look in the second list (just above) and see that 00185892 (same as 185892) contains acceleration, as it should.

Use of this cross-referencing method makes it very easy to find the synonyms of a given word. First the word is found in the noun file (e.g. acceleration) . Then each of its sense numbers is found in the sense file (3524146 185892 9175297). If the word the user entered is in any of the lists of nouns that are synonyms of the first word, then the user would get the question correct.

Upon entering a sentence, the program checks the content of the user's sentence against the master sentence content to determine whether the correct answer was given. However, a problem arises when the user's sentence contains a noun which is a synonym of the "correct" noun in the same sentence position. In the earlier versions of the tutor such a sentence would be marked "incorrect".

```
PUT THE TRASH IN THE TRASH CAN ->
   CS_1: [CAUSE YOU [GO TRASH [TO [IN TRASH CAN]]]]
```

```
PUT THE TRASH IN THE GARBAGE_CAN ->
    CS_1: [CAUSE YOU [GO TRASH [TO [IN GARBAGE CAN ]]]]
```

In the current system, the synset for one sense of 'trash' would intersect the synset for 'garbage', and the answer could be scored correctly.

A complete list for all of the nouns contained in Wordnet was compiled – it contains 87,566 nouns. In order to be useful for the current matcher though, the synonyms that are discovered must be translated into either Arabic or Spanish, which are the languages used in the MILT. In addition, an authoring facility must be added to allow instructors to add new words to the matcher. Let us consider these tasks in reverse order.

The specification calls for storing the synonym sets in 2 forms. The first form, which would be used on-line in the matcher, would contain a subset of the synonyms that have been deemed relevant for a given lesson, and would be compiled out from the master Synset list. The reason for doing this would be to create a smaller data structure for on-line use to speed up the search process for the synsets. The full synset list would be stored as a separate file, which could be accessed by the Editor. As mentioned above, this editor cannot require any background in either Linguistics or Computational Linguistics, or require computer programming by the user.

The designed editor would use a graphical user Interface to prompt the user. The user could enter a chosen word on the screen. The program would check to see whether the word is already in the lexicon and inform the user if it finds the word's synset. If not, the user would be prompted to enter a synonym, or category that the word is a particular instance of. For example, the user may want to enter ' Sherman Tank' and have 'tank' as a synonym. A list of synonym sets for the word would then appear. In this case, the syn-sets taken form Wordnet would be:

```
sense1: tank army_tank armored_combat vehicle / an enclosed armored military
vehicle; has a canon and moves on caterpillar treads
sense2: tank storage_tank / large ( usually metallic canister for holding gases or
liquids).
sense 3: tank tankful / as much as a tank will hold
sense4 : tank_car tank /freight car to transport liquids or gases in bulk
sense5: cooler tank / cell for violent prisoners
```

These synsets would each be displayed in a separate window. The user could click on the appropriate window. The synset associated with this sense of the word would then be added as synonyms to the new word. This process could be repeated for cases where a new word has more than one sense, or new synonyms could be tried if the user feels that the first synonym did not include an appropriate sense for the new word. An example of the editing screens are given below:

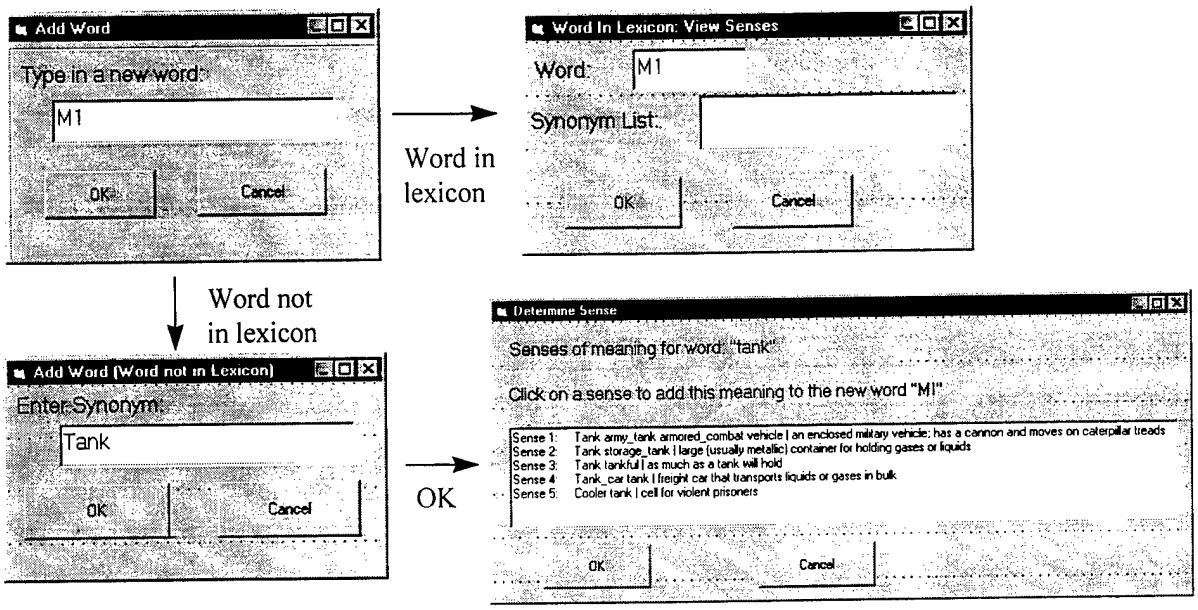


Figure 2. Synset editing screens

The tutor would then be able to use the new word, now appropriately defined, as if it had already appeared in the lexicon. In addition to being used by the MILT, this utility would be used in the dialogue system for military skill tutoring described below, enhancing the ability of this system (which is in English) to understand if the student is on the right track and to be more flexible and realistic about accepting student input. To be useful for MILT however, this system must be translated into the appropriate languages of instruction in the tutor. The results of the synset process must be translated into either Spanish or Arabic. This process, whether performed by an Arabic or Spanish linguist, or by the end user would be performed under the guidance of the lexical editor.

The Alpnet lexicon, which is the lexicon used for Arabic contains 37,000 lexical entries. All of these entries contain English glosses. The Spanish lexicon contains approximately 30,000 words. The words in both of these lexicons contain English glosses. The nouns would be extracted from the Spanish and Arabic lexicons and associate them with the synsets of their English glosses. As is well known, polysemy does not carry over in regular ways across languages. Therefore, these derived synsets would have to be checked by a native speaker, and where a word that is polysemous in English, matches to only one synset in Arabic, or Spanish that set would be deleted by the native speaker linguist. If a word was polysemous in either of the target languages, in a way that was not true of English, the system would experiment with adding another synonym from English. The same editor used to add synonyms by the user could display the synsets in a convenient way for the Arabic or Spanish linguist. That is, all of the synsets associated with the English word would be displayed, and the Arabic or Spanish linguist would simply click on which ones were appropriate, or add synsets. Basically the same procedure is used by the end user. The Spanish and Arabic databases created by the Arabic and Spanish linguists have Arabic or Spanish words and their glosses associated with the revised synsets relevant for the language. The user would then type in the relevant synonym for the target word in the relevant language and then choose among the appropriate synsets as described above.

The final subtask would be to provide the appropriate morphological class for added words, so that they could be correctly decomposed by the morphological analyzer and retrieved in the lexicon. This is important because the roots in this lexicon need to be stored, but since both Spanish and Arabic are heavily morphologically inflected, the appropriate roots must be found, given the inflected forms.

For example, nouns, in Spanish fall into three broad morphological classes. In the interface, examples of each case would be presented and the user asked which one the noun behaves like:

MASCULINO		FEMININO		MASCULINO/FEMININO	
M1=O-OS	jugo, jugos / acto, actos / abuelo, abuelos	F1=A-AS	arena, arenas / águila, águilas / abuela, abuelas	MF1=O-OS-A-AS	zapatero, zapateros, zapatera, zapateras
M2=NOT O-OS	ser, seres / tema, temas / poema, poemas /ambiente, ambientes / treinta, treintas	F2=NOT O-OS	ciudad, ciudades / amplitud, amplitudes / gente, gentes	MF2=R-RES-RA-RAS	boxeador, boxeadores, boxeadora, boxeadoras
etc.					

Table 2. Morphological noun examples

After seeing examples of nouns, users would have to click on the appropriate class. The system would ask the user to provide a singular form for the word in question and to tell us whether it is masculine or feminine. The user would also be asked to give the plural. This is less important for Spanish, where plural formation is regular, but it is crucial for Arabic because the system cannot predict when s form has a regular or a broken plural.

Then Kimmo would know how to build the morphology for the noun. Beyond this point, the user would need to provide an English gloss. Similar help would be provided to users to have them tell us which conjugation class to look for in decomposing Arabic nouns

In summary, the process for adding these nouns to the lexicon would involve the following steps:

1. Build noun sense classifier and synset databases for English. This has been accomplished during Phase I of this contract.
2. Build synset editor
3. Using glosses in ALPnet and the Spanish dictionary, translate noun sense databases and synsets into Spanish and Arabic
4. Build capacity to precompile subsets of the classifiers and databases for use during on-line processing
5. Install precompiled subsets into MILT. Add full Arabic and Spanish systems and the authoring tool into authoring component of MILT. Build mechanism to precompile different or additional subsets of large databases for on-line use.

Task 4: Develop a Dialogue System

Development of an authorable dialogue system is a complicated procedure involving several different tasks. The first step would be to pick a representative domain and target scenario that is of interest to the military community, has commercial application, and could be used as a basis for authoring other similar domains. In order for a system to engage in a dialogue, a minimum level of information must be provided to the system concerning what is acceptable dialogue. Computational linguists and AI knowledge engineers have at their disposal sophisticated tool-kits -- the formalisms of

their fields -- that enable them to capture regularities at an abstract level, and thus to provide the needed information to the system without specifying a lot of detail. However, one must be a trained computational linguist and knowledge engineer to use these formalisms. The layperson-author, who is not trained in these formalisms, must use a different set of tools that do not capture the same regularities as the linguists' and knowledge engineers' tools. Hence, although it is possible for untrained authors to provide a computer system with the same minimum level of information required for dialogue, these authors would necessarily do so at a more concrete level of description, one which requires explicit specification of the possible directions a dialogue could take. This unavoidable tradeoff is reflected in the decision to use schematic dialogue modeling technology in the system design. With this technology, laypersons would be able to author dialogue-based systems by providing examples of utterances for both participants, and by organizing these examples in visually presented networks.

During Phase I, an emergency medical treatment scenario was chosen based upon the above criteria. The domain was discussed under Task 1. Training in medicine was chosen as an example topic because this is an area in which the military has training needs. However, the dialogue techniques outlined are not specific to medicine, nor to training dialogues. In fact, they were designed primarily to meet the requirement of authorability, not specifically to model tutorial dialogues. The dialogue modeling techniques used are applicable to any type of dialogue in which authors could foresee most reasonable courses of dialogue, and specify these with examples and with visual diagrams. It is believed that this includes many mission training dialogues. Any limitations of the kind of dialogue that could be modeled are due to the requirement that this be a layperson-authorable system. In this section the conceptual dialogue system specification is described.

Context Sensitive Responses and Global Coherence

The most primitive computer-based training systems map single, literal user inputs to single, literal responses. Early natural language systems such as ELIZA (Weizenbaum 1966) used patterns for response rules, enabling a slightly greater variety of inputs to be associated with a given response. However, semantic and dialogue regularities are not captured by this syntactic pattern matching formalisms. In theory, if one wrote very many rules, organized them into different rule sets for different contexts, and exerted control over which rule set was active at a given time, one could write interactive systems in this sort of syntactic pattern matching. However, authoring at such a superficial level would be tedious in the extreme. A huge number of well organized rules would be required. The REAP and LCS technology described in this report addresses this problem for the left hand side of input/response rules. This technology would enable us to write rules that match to a much broader variety of syntactic variations of input with a given meaning. Hence it would provide the user with great flexibility in expression.

However, there is still a lack of flexibility to be addressed: flexibility of response. The system needs to be enabled to give different responses in different situations, varying its response to the same input according to factors such as estimates of the user's knowledge, the user's and system's goals, and, in cases where the system has a particular agenda to follow, the state of the dialogue with respect to that agenda. For example, the same input may need to be mapped

```
"His left leg is bleeding."
```

to different responses depending on the situation, e.g., whether or not the leg is bleeding; whether or not the patient is breathing, has a pulse, etc.; and whether the system is expecting the patient to treat the patient. More control is also needed than local input-output rules could give us, however flexible these rules are. Pedagogical strategies need to be expressed for teaching using multiple coordinated interactions with the student -- i.e., globally coherent discourse.

In this section, plans for a natural language tutorial dialogue system are described that address these concerns. Begin with simple "core" functionality that is embedded into more complex functionality. The choice of technology is strongly driven by the need for an authorable system. The core

functionality would be deliberately simple, as is necessary in order to meet the design goal of authorability by nonspecialists (non-linguists, non-programmers). More sophisticated functionality, which places greater demands on authors, would be obtained by embedding this core functionality in a larger context. Throughout, the various components of dialogue strategies that could be authored are discussed. The section concludes with some examples of what authoring interfaces might look like.

Preview of Discourse Model

The discourse model would be a layered model, including the following layers:

- *pedagogical styles* - global preferences and strategies that define interaction styles
- *dialogue models* - the "strategic" structure of multiple exchanges
- *exchange models* - the "tactical" aspects of how to initiate and complete exchanges
- *discourse goals* - intentions to affect the hearer's mental state in certain ways
- *rhetorical acts* - typical patterns of utterances that could be used to achieve intentions
- *communicative acts* - individual utterances

In the following sections how the system would generate single replies by mapping discourse goals to communicative acts is first described, followed by how this functionality would be embedded in the extended dialogue and exchange models. The various models will be explained and exemplified as they are encountered.

Planning Single Replies

The core functionality of the dialogue component is the ability to generate multiple utterances that might achieve a given discourse goal, and the ability to select from these multiple possible responses the one that best fits the current situation. It is called the "core" functionality because, taken alone, it is sufficient to implement a simple "reactive" dialogue system, i.e., one in which the system only responds to queries of the user, and never changes its response strategy (though it may change responses to the same query as the context changes). This functionality has three components:

- *response rules* for mapping user input (in LCS form) to an intention to respond in a certain way;
- *refinement rules* for generating possible utterances that satisfy this intention; and
- *response selection preferences* for selecting from among those utterances the best one for the current situation.

The core functionality would be enhanced by a representation of context, which consists of the dialogue history, the practice session state or history, and a student model. A number of abstractions are used to model the planning of single replies. None of these are specific to pedagogical domains. Illocutions and rules pertaining to different ways of explaining and describing were used because they have been developed for pedagogical applications. However, rules could be written for a full range of illocutions and speech acts. For example, authors could be provided with the choice of different ways of conveying information, such as asserting, predicting, conceding, disputing, or suggesting; with different ways of directing, such as requesting, commanding, or prohibiting; with promises and offers; and with different kinds of acknowledgments. Discourse resources could be provided for convincing someone to do something as well as to believe or understand something. These resources could be placed in service of other kinds of dialogue in addition to pedagogical dialogues.

Response rules.

The response rules instead map an input LCS to a *set or sequence of "discourse goals."* Discourse goals are intentions to have certain kinds of effects on hearers. One could achieve discourse goals by performing *"rhetorical acts"*, which are particular kinds of explanations and descriptions (Moore and Paris 1993; Suthers 1993). For the convenience of authors, the system would allow authors to map inputs to directly to rhetorical acts as well as to the more abstract discourse goals.

For example, suppose the system needs to be able to respond to a query concerning whether the patient is bleeding. One could write the following response rule, mapping the user's query via LCS to a discourse goal:

```
((2 (CASE NOM) (PERSON THIRD) (GEN F M) (NUM SING) (THETA AG) (HUMAN -)
   (GLOSS "he") (SURFACE "he"))
 (1 (ASPECT) (GLOSS "bleed") (SURFACE "bleeding") (VOICE ACTIVE)
   (SURFACE "is") (GLOSS "*root = be") (HUMAN -) (NUM SING) (GEN F M)
   (PERSON THIRD)))
   --> (believe ?hearer (attribution ?patient breathing-status ?status))
```

The variables ?hearer and ?patient would be bound by the context, while ?status would be bound by matching to the knowledge base. Alternately, an author could choose to specify that a given question be addressed using a given rhetorical act, which consists of a kind of explanation and a set of one or more propositions:

```
--> (describe (attribute ?patient breathing-status))
```

In this expression, "describe" is the kind rhetorical act, and "(attribute ?patient breathing-status)" is a way to access the value of the patient's breathing status (attribute is called an "accessor function"). Other kinds of rhetorical acts -- called "rhetorical illocutions" -- include Define, Describe, Compare, Illustrate, etc. (Figure 3). See Suthers (1993b) for full details on rhetorical acts, accessor functions, and many other aspects of the present design.

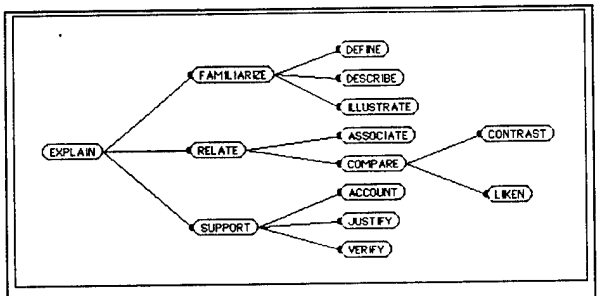


Figure 3. Rhetorical acts

One could also simply direct the system to perform a particular communicate act, such as informing the user of the patient's status:

```
--> (inform (truth-value (attribution ?patient breathing-status breathing)))
```

In this expression, "inform" is a "speech act," which affects how the generated text expresses an attitude towards the expressed information, such as what the user is to do with the information (in this case, believe it; Bach & Harnish 1979). "Attribution" is a predicate, the counterpart of the function "attribute".

Discourse goals and rhetorical acts are abstract -- they could be realized in text in multiple ways. This is important because it means the response rules do not constrain the possible responses unnecessarily. Variation in responses is important when the appropriate response depends on the hearer and the context. The mechanism described in the next section generates the possible responses.

A response strategy consists of a set of response rules. In a simple system, this set might never change. In a more sophisticated system, different sets of response rules would be invoked for different contexts, such as portions of an interrogation or tutorial intervention.

Authoring response strategies.

The authoring interface for response rules must provide the user with facilities for writing LCS patterns for the left hand side, and discourse goals (etc.) for the right hand side. The system would facilitate authoring of LCS patterns by nonspecialists using parsing, unification, and machine learning techniques. The author would enter a number of sentences which the response rule should match, and

others that are similar in appearance but should not match. The authoring tools would parse the sentences into LCS, and use LCS unification to find the most specific LCS pattern that matches all of the parsed positive examples. Using standard machine learning techniques it would adjust the pattern to exclude the negative examples (Mitchell 1977). In this way, the author need not ever write or even see LCS syntax directly.

Authoring of the discourse goals, rhetorical acts, and communicative would be facilitated by menus of (a) illocutions, (b) predicates and functions, such as "attribution" and "attribute" respectively, and (c) domain terms. A meta-menu would embed these menus in examples of typical, legal patterns. For example, one could select one of these meta-menu items:

```
(<discourse-illocution-menu> ?hearer
 (attribution ?patient <domain-term-menu> ?value))
(<rhetorical-illocution-menu>
 (<predicate-menu> ?patient <domain-term-menu> <domain-term-men>))
(<rhetorical-illocution-menu>
 (<function-menu> ?patient <domain-term-menu>))
```

and then fill in the templates by selecting from the submenus. Once the response rules are defined it is easy to author response strategies

Refinement rules.

Given that the system has decided it should achieve a certain discourse goal (or perform a given rhetorical act), it needs ways to refine these into utterances. For example, suppose there is a patient with kidney injury, and the student asks

```
"What are kidneys?"
```

Matching a response rule might yield the discourse goal that the hearer know about kidneys.

```
(know ?hearer kidneys)
```

This goal must be refined into acts that can achieve it. Refinement is accomplished with refinement rules, which consist of dominator expressions, refinements, and constraints. For example, the following rule says that one can get the hearer to know about a topic by performing a "familiarize" rhetorical act:

```
Know by Familiarizing:
        Dominator: (know ?hearer ?topic)
        Constraints: none
        Refinement: Familiarize(?topic)
```

Dominators say what the rule would achieve. These are matched to the current discourse goal or rhetorical act to identify rules that apply. Constraints perform other tests to determine whether the rule should apply. Refinements provide one way to achieve the rhetorical act. A refinement can itself be a rhetorical act, or it can be a communicative act.

Continuing our example, there are different ways to familiarize one with kidneys. For example, one could describe them, or draw an analogy between them and something that is more familiar to the hearer. Descriptions themselves could take different forms, such as describing their function, their structure, or their anatomical context. These different forms of descriptions could be mixed. It is the job of the refinement rules to generate these possibilities.

```
Familiarizing by Analogy:
        Dominator: Familiarize(?topic)
        Constraints: (know ?hearer Analog(?topic))
        Refinement: Liken(?topic, Analog(?topic))

Familiarizing by Describing:
        Dominator: Familiarize(?topic)
        Constraints: none
        Refinement: Describe(?topic)
```

```
Describing a Device by its Function:
        Dominator: Describe(?organ)
        Constraints: Subsumes-p(Device, ?organ)
        Refinement: Describe(Function(?organ))

Describing an Object by its Structure:
        Dominator: Describe(?organ)
        Constraints: Subsumes-p(Object, ?organ)
        Refinement: Describe(Structure(?organ))

Describing Referents of a Functional Expression:
        Dominator: Describe(?function(?entity))
        Constraints: Subsumes-p(Accessor-Function, ?function)
        Refinement: Inform(Referents(?function(?entity)))
```

One possible response is generated by chaining the rules Know by Familiarizing, Familiarizing by Describing, Describing a Device by its Function ("organ" being a kind of "device"), and Describing Referents of a Functional Expression. "Inform" is a communicative act -- it can be directly executed by the system. "Referents" is analogous to eval in Lisp: it evaluates the functional expression (e.g., "Function(kidney)") to get its value (e.g., "blood filter").

It is possible to write refinement rules that refine a discourse goal or rhetorical act into multiple utterances. In the core functionality, the sequence in which these utterances are realized is pre-specified. Explicit reasoning about ordering of utterances is possible, and would follow Suthers (1993a) if this path is pursued.

Authoring refinement rules.

The conceptual design breaks the job of planning a response into small, tractable pieces. These pieces could be reused to solve different planning problems. This has significant advantages for authoring. A graphical interface enabling authors to edit networks of refinement relations formed by collections of rules such as the above would be constructed.

Refinement strategies consist of sets of refinement rules. As with response strategies, it is easy to collect refinement rules into different refinement strategies and change these strategies as needed. A major advantage of the separation between response and refinement rules is that one could change one without changing the other. For example, the above rules for familiarizing, describing, etc. might always apply, while the intention to reply in a certain way to a certain class of utterances would change throughout a tutorial dialogue.

Response selection.

Even with the small rule set above, three different responses to the question "what are the kidneys" could be generated. The number is actually greater, because the "Referents" evaluation process can result in multiple values for the functional expression. For example, a knowledge base may have several functional descriptions of kidneys, appropriate for different kinds of users and purposes. The next step in the core functionality is to select from the possible responses a subset that would actually be uttered. This selection should be done on the basis of contextual factors provided by other parts of the system.

Contextual selection is handled by ordered or weighted sets of preferences (Hovy 1990, Suthers 1993a). A preference is a function that would take two possible responses and return one of three values: >, =, or <, depending on which argument is preferred, if either. For example, an important preference is "say something new." This compares proposed responses to the dialogue history, preferring those which do not repeat previously given information. Other preferences include preferring responses that minimize the number of concepts unfamiliar to the user, that minimize the number of propositions in a response, and that use "reference examples" -- examples which form a common anchor for many situations and concepts. Preferences can conflict with each other.

A simple algorithm exists to apply a preference to a set of alternatives, partitioning them into a sequence of subsets where the first subset contains the most preferred alternatives, the next contains the next most preferred, etc. (Suthers 1993a). Alternatives in the same set are equally preferred. If there is more than one preference, the system can arbitrate between them either by priority or by weighting. In the priority scheme, one preference has its say first, generating a most preferred set. If there is more than one alternative in this set, the next preference is applied to it, and so on, successively filtering with preferences until one alternative remains or a random choice must be made. In a weighting scheme, all preferences are applied, but they are given weights, and a linear sum is computed. The best approach would be determined empirically.

For example, suppose refinement produced two alternate communicative acts for familiarizing the hearer with kidneys. One might describe its function in complex medical terms, while another might say it is "like a filter". If the user model indicated a lack of understanding of medical terms, the simpler explanation would apply.

Authoring response selection strategies.

A response selection strategy, then, consists of either an ordered sequence of preferences or a set of weighted preferences. A system could change its selection strategy by changing the sequence, or the weights. Individual preferences can be complex to author, as they must extract components of the proposed responses and compare them to data structures such as the dialogue history and the user model. Fortunately many preferences need only be authored once, as they are completely general with respect to domains. Response selection strategies, on the other hand, are easy to author: one simply selects named preferences, and either puts them in an ordered sequence or adjusts weights on them.

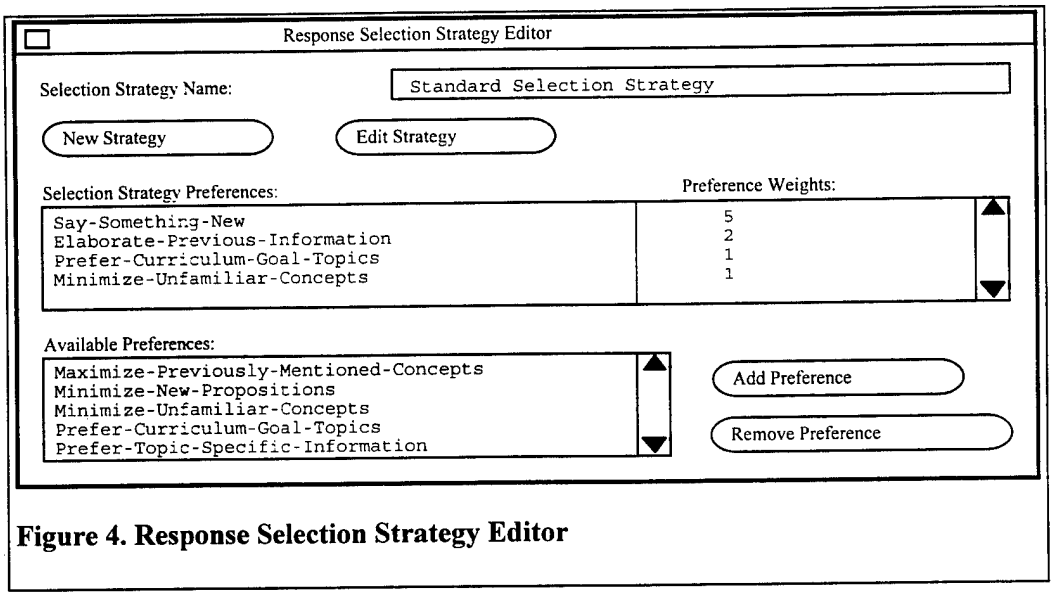


Figure 4. Response Selection Strategy Editor

An optional facility is the ability to specify how the system should choose between multiple responses that serve the same goals. The dialogue planner would be capable of generating multiple responses to the same question. A preference-based mechanism is used to select the response most appropriate for the current situation. If the author elects to control this selection, an editor such as that shown in Figure 4 would be used to select and weight the various preferences by which one makes the choice. (Preferences themselves would be pre-defined. Being highly portable between application domains, it is not expected that much authoring of preferences will be required.) Response selection strategies would be associated with strategic states. An author can elect to ignore this dimension of flexibility, and leave selection up to a pre-defined response selection strategy.

Summary of core functionality.

To review, the basic functionality has now been defined for selecting an intention to respond to an input utterance in a certain way, for refining that intention into possible utterances (abstractly defined), and for selecting between these alternatives. Figure 5 summarizes this functionality. Authoring would take place in the following ways:

- Resources
- Response Rules - These map LCS to discourse intentions and/or rhetorical acts, and must be authored for new domains.
- Refinement Rules - These map rhetorical acts to communicative acts. They might be edited for new domains, but many rules would re-apply.
- Response Selection Preferences - These filter communicative acts. Although difficult to author, little or no authoring is expected for new domains, provided proper abstractions are used to write the preferences.
- Strategies(all are easy to author given the above resources):
- Response Strategies - Unordered sets of response rules.
- Refinement Strategies - Unordered sets of refinement rules.
- Response Selection Strategies - Ordered or weighted sets of preferences.

This basic functionality would need to be embedded in a system that has a broader view of the dialogue.

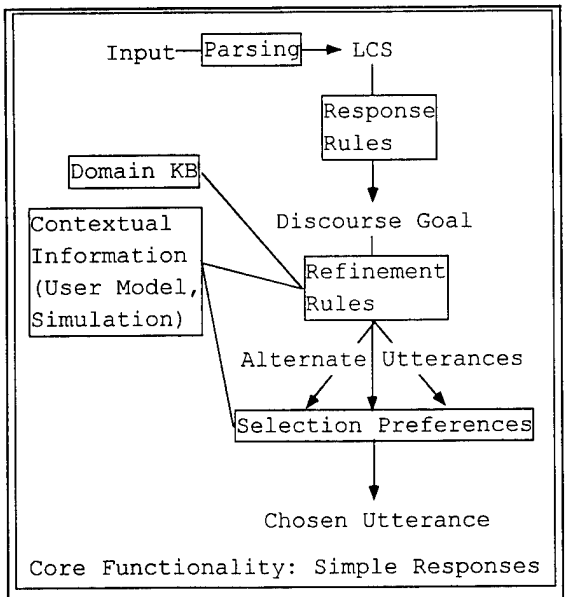


Figure 5. Planning simple responses

Dialogue Management: Changing Strategies

The core functionality provides local response strategies. With the addition of basic handling of follow-up questions (e.g., as in Moore 1995), the system would have sufficient functionality to support simple, user-initiated question answering. The remaining design is intended to support system-initiated and controlled dialogues in which the system is asking the user questions, and generally setting the agenda. Combining the two with an interrupt mechanism and context stacks, the system would be able to handle a mixture of system-controlled dialogue and user-initiated questions that interrupt the intended course of that dialogue.

Our approach to dialogue management is driven strongly by the need for an authorable system. The technology used is not intended to represent the state of the art in theoretical models of dialogue. Rather, it is intended to support a broad range of flexible dialogue to the extent possible while also using control structures that map easily to visual representations that nonspecialists can easily understand and modify. Hence for this design, schematic formalisms for discourse structure were chosen.

There are two issues to address: (1) representing and controlling the execution of possible trajectories through a tutorial dialogue agenda; (2) changing the response strategies as a function of location in that agenda. These issues can be separated by using a hierarchy similar to the "discourse management network hierarchy" of Woolf and McDonald (1984). The present design consists of three layers: tactical exchange networks, strategic dialogue networks, and pedagogical styles.

Pedagogical styles.

Pedagogical styles are states that specify the default response strategy, refinement strategy, and response selection strategy to use. (Other default strategies and other parameters may also be added as the system functionality is extended, for example to indicate how easily a system would give an answer away when a student is having trouble.) Pedagogical styles are simply abstractions used to collect together useful strategy configurations, so that these configurations may be conveniently referenced. Once the component strategies are authored, authoring of a pedagogical style is a trivial matter of selecting one strategy of each type.

The collections of global preferences and default response strategies which can be termed "pedagogical styles" are not specific to pedagogy. They are better termed "interaction styles." Regardless of name, they help layperson-authors capture information common to a type of discourse. Although the design planned to use these abstractions to capture different types of pedagogical interactions, they can be used to capture aspects of other kinds of discourse as well. For example, a soldier conducting a dialogue with a commanding officer, a subordinate, or a prisoner would phrase utterances differently for each of these interlocutors. (These different styles of discourse are related to the linguistic notion of "register," although our interaction styles serve a specific control role in the system that is not identical to "register.")

Tactical exchange networks.

Tactical exchange networks represent stereotypical, self-contained sequences of interactions that achieve certain pedagogical purposes, for example, to compare a student's performance with that of an expert, or to quiz the student on a particular fact.

A tactical network consists of a number of states, and transitions between them. Each state references a response strategy (set of response rules), here called a Local Response Strategy and organized in a table. The response rules are extended: in addition to a discourse goal or rhetorical act, each rule consequent specifies a *response category.* These serve to categorize the user's input into abstractions appropriate for controlling transitions to new states. These categories can be invented to suit the application, but typically in an interrogative system would include "correct," "incorrect," and "change of topic". In addition to the local response strategy, there would be one global response strategy -- that specified by the pedagogical style. The purpose of the global response strategy is to provide default ways to respond to user input that is not specific to a state. Local Response Strategy tables look like:

user input	intention	category
[LCS1]	RI1	correct
[LCS2]	RI2	near-miss
[LCS3]	RI3	incorrect

Each state would also reference a state transition table that maps response categories to states (either the same state, or others). Optionally, the transition table can also specify transition to a new pedagogical style. Finally, one can write degenerate states that have no response strategy and no

transition table, only a rhetorical or communicative act and a next state. These are used to generate unconditional utterances. State transition tables look like

category	new state	pedagogical state
correct	LRS2	Review
incorrect	LRS3	Remediate

Tactical networks are entered at a specified state. The user's input is matched against the LCS left hand sides of the entries in the Local Response Strategy. When a match is found, the core response functionality is invoked to plan an utterance based on the discourse goal or rhetorical act in the right hand side of the response rule, the utterance is made, the response category (based on user's previous input) is used to determine the new state, the user's next input is read, and the process repeats.

State transition tables can invoke other tactical networks, in which case a stack is used to store the state of the current network before entering the new one. This facility is for convenience of expression of locally recursive dialogue structures, and for handling interruptions.

Tactical networks are used for local interactions, of just a few utterances. For example, a tactical network might say how to ask the user a question about a fact, and give feedback according to the correctness of the response. Another might indicate how to answer a question with handling of follow-up questions. Interruptions are handled as follows. If a user input LCS does not match any local response strategy entry, the global response strategy is tested. If a match is found, the input is considered an interruption or change of topic. Using other tactical networks one can easily write response strategies that choose to either defer the interruption or to process it. In either case, the stack mechanism assures return to the prior dialogue. There are limitations in this mechanism. In particular the interruption could make some of the pending (stacked) dialogue obsolete (Grosz and Sidner 1985). These issues would be dealt with in extensions to the system.

Tactical exchange networks represent stereotypical, self-contained sequences of interactions that achieve certain superordinate intentions. In this report these are characterized as achieving "certain pedagogical purposes," and used pedagogical examples. It is more accurate to characterize them generically as achieving "superordinate intentions" because, like the other modeling tools used, they are not specific to pedagogical dialogues. Although pedagogical "response categories" are used (e.g., "correct" and "incorrect") in our examples of the "local response strategies" that define tactical network states, these categories are arbitrary, and can be redefined as appropriate for other dialogue types and topics.

For example, an exchange network can be used to represent possible sequences of interactions that result when I ask you for a given item of information. You might respond by telling me the information that I requested, in which case the exchange network exits with success; by asking for clarification, in which case the system enters a clarification network; or by refusing, in which case the network might either exit with failure or specify how to engage in further attempts to obtain the information. This collection of related, stereotypical patterns of interaction could be applied to different situations, such as asking someone for the time or for directions, asking for troop location, or asking a patient what hurts. (Interaction styles would take care of some of the differences in how one goes about these dialogues, although some re-authoring for new domains may be required as well.)

Authoring tactical exchange networks.

Authoring of tactical networks would be facilitated by a direct mapping of the transition tables to diagrammatic structures. States would be displayed as shapes such as boxes for the degenerate single-utterance states, and diamonds for normal states (indicating that a decision is made). Arcs emerging from the shapes are labeled with the response category that selects that arc. By double-clicking on a node, one can display its internal structure: its local response strategy table and its state transition table.

Figure 6 shows how similar state transition diagrams could be used to model discourse structure at a finer granularity. Tactical exchange networks indicate how discourse goals (and the utterances that

achieve them) are organized in service of the superordinate goals of the application domain. Response categories (e.g., "conscious," "granted," and "explain") control transitions between tactical states, and exit parameters (e.g., "may-treat") communicate control information to the strategic network. As with strategic networks, tactical networks with have hyperlinks to relevant components, and can be "run" for testing once their states are at least partially defined.

Double-clicking a tactical state would open a tactical state editor (Figure). (Strategic states would have similar functionality.) The description has now descended to a level of detail that is best represented using lists of reusable resources rather than diagrams. A tactical state makes use of a list of response rules, which say how to respond to particular interlocutor utterances. Each rule is also associated with a user-defined "response category" which is used to control movement in the tactical network, as shown in Figure . (Response categories not associated with an arc in the network retain execution within the same state.) Authors can optionally specify an utterance to be generated unconditionally upon entry into the state. Once one or more response rules are selected, the author can run test trials of the state. The authoring tool would help the user identify missing state transitions.

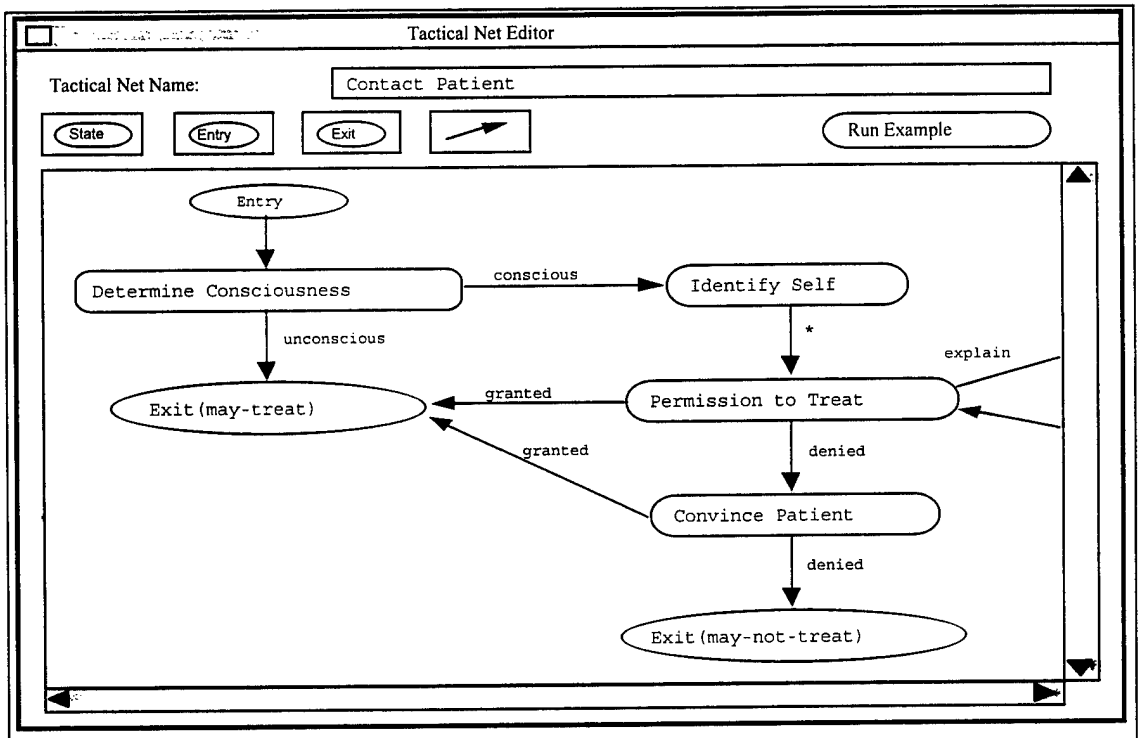


Figure 6. Tactical Network Editor

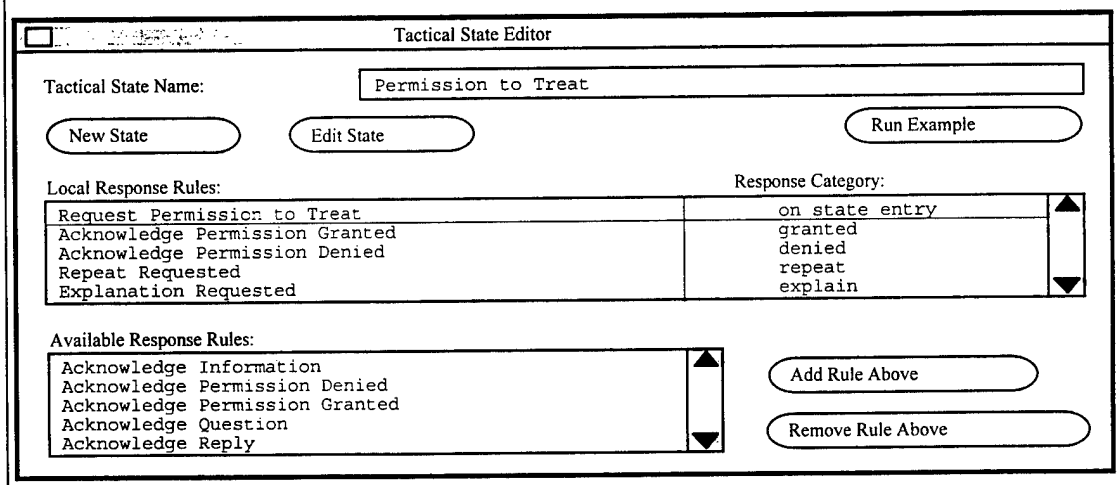


Figure 7. Tactical State Editor

Strategic dialogue networks.

A strategic network is similar to a tactical network, except that it organizes tactical resources and is conditioned on the user model. Strategic Response Strategies contain pointers to tactical networks and simulation resources, instead of rhetorical intentions. Their state transition tables are more likely to include changes in pedagogical state.

Strategic dialogue networks would be used to organize the introduction of the practice sessions and the reflective self assessment dialogues. Each instructional system would have one strategic network for a given instructional unit. System execution begins at a designated state in the strategic network. This

state should be authored to invoke tactical networks that perform the appropriate introductions to the task at hand. The network then enters a state that invokes the practice simulation. The user model is updated by resources within the simulation, which assess how the student performs on each step of the procedures and/or each component skill. Questions asked during the practice session are handled by a tactical exchange network designated for the purpose. When the student has completed a practice session, the remaining strategic network controls the reflective follow-up dialogue in which these results are reviewed with the student.

Our preferred pedagogy is for the system to ask the user to self-assess before reporting the system's assessment. This provides the student with self-assessment skills, important for internalizing the ability to learn without the help of the system. The system would ask the user a question concerning the user's performance; the user would answer; and the tactical network would reply according to whether the user's self-assessment were accurate. The dialogue and authoring resources provided would also make it possible to write more "instructionist" styles of interaction, for example, quizzing the student on certain points of fact, or telling the student when he or she is wrong.

High level dialogue models are represented using strategic dialogue networks. These networks enable the layperson-author to organize dialogues according to the flow of superordinate intentions in the dialogue -- that is, how the goals of one interlocutor change as a function of the other interlocutor's utterances. Each dialogue network corresponds to one session with the system, whether simulated prisoner, patient, or tutor. Each state in a dialogue network corresponds to a goal or set of related goals held on the part of the simulated interlocutor. These strategic states organize tactical resources (to be discussed next) that are relevant to these goals. This functionality is emphasized in this report with a pedagogical example: dialogue networks control when the system introduces a problem, when the student practice the problem, and when the system evaluates the students' work. However the utility of dialogue models is not limited to this kind of dialogue. For example, one could just as well use a dialogue network to model a simulated prisoner. In this application, states in the network could be used to represent the prisoner's intentions to cooperate or not cooperate with certain kinds of requests. Each state would contain the tactical exchange networks that indicate how the prisoner responds to particular questions with particular utterances, in a manner reflecting these superordinate intentions.

The foregoing sections illustrate the bulk of the authoring required for dialogue capabilities in new domains. The "refinement rules" are expected to port well between domains. These rules indicate how to take a reply pattern and translate it into an LCS specification for text. Moderate additions may be needed for particular ways of communicating in specialized domains. In this case facilities such as a graphical display of the refinement hierarchy and reply pattern menus for authoring would be provided. Response selection preferences can also be authored for new domains, although it is expected that this will be rarely required.

Generating Natural Language

The result of the above process is one or more communicative acts (illocutions such as "inform" applied to propositions). These would be realized into natural language using technology such as FUF (Elhadad 1992). This technology gives us a set of templates based on discourse variables like speaker intention. The actual choice of the words to use and the syntactic frames to use to express the propositions indicated by the demands of the dialogue planner would be done through an LCS generation facility.

The first task in developing a natural language generation (NLG) system is to design the appropriate representations for the various levels of the computation. In this NLG system, the Minimalist framework is followed in that it is using only the bare minimum of representational levels: a lexicon, a semantic or "logical" form and a surface form which would be the output of the system. As the semantic representation is already given to us by the parsing system, only the lexicon and the surface representation need to be designed.

The surface representation is fairly straightforward; while a few modifications may be needed, the general structure used by the parsing system can be easily adapted to the generation system. In developing the lexicon, however, there are difficult issues to address in order to have a fast real-time system. The lexicon in this system would contain thousands of entries, and in order to find appropriate entries given an LCS from the parsing system, there needs to be a sophisticated indexing system based on the LCS structure.

This lexical structure would be implemented by keeping an "index" tree stored in memory which would be searchable via LCS components (such as EVENT, FIELD, THING, etc.). A link between two nodes would characterize the addition of exactly one additional LCS element. Each LCS node would contain a key-word to access the appropriate lexical entry stored in a database on disk, where the program may retrieve any relevant information about that particular word.

In Phase I the portion of the system that would take an LCS structure from the parser and search the lexicon for the appropriate words using only this indexing system was worked on. Adding the decisions made by the grammatical component of the system as an information source here may be considered as well --- the aim being to further constrain our search space with the restrictions of the grammar in mind.

Once a satisfactory lexical subsystem is in place, the development of the grammatical subsystem would begin. Again, the conceptual system was designed with the principles of Minimalism in mind. In Minimalism, only two operations in the grammatical derivation of a sentence would be used: merge and move. The merge operation would simply combine a word or phrase with the current derivation. The move operation would copy a word or phrase in the current derivation to the front of the structure and then delete the old occurrence of that word or phrase. These operations are constrained by restrictions of "local optimality." That is, the system would avoid doing anything unless it is absolutely necessary to do so. This aids in reducing what is perhaps the greatest problem encountered in NLG: overgeneration. In addition, the notion that constraints are local reduces the amount of computation, as the system would only consider those aspects of the structure which are "close" to the operation being considered. This allows the building of a licensing structure to generate answers in a maximally efficient way.

Authorability

Within the domain chosen, there are several different kinds of knowledge that must be embedded in the tool and skills that a student may require training practice on. Examples are:

Human Anatomy and Physiology: Declarative knowledge of anatomy and physiology, used to answer questions, and to provide an ontology for dynamic models.

Trauma and Disease: Declarative, categorical knowledge of different types of trauma and disease, in terms of their effects on anatomy and physiology.

Medical Interventions: What kinds of things can a practitioner choose to do?

Physiological Simulation Model: Transitions (possibly probabilistic) between patient states as a result of default physiological functions, trauma or disease, and/or medical interventions.

The conceptual system would demonstrate that the authoring tools would support movement into several different medical domains, including Trauma Care and possibly Dermatology.

Programmers and instructional designers would be able to make derivatives of the NLP medical system. These derivatives would account for most of the tutoring strategies for teaching in the medical domain. Several examples of dialogues generated by the system can be displayed to demonstrate the generality the knowledge base. It could be demonstrated that the knowledge base contains generic instructional strategies for teaching in the medical domain and this could be tested empirically with a number of users and domains. It is expected the tools described to be highly usable without sacrificing tutoring systems power.

Authors would develop courseware without having to encode instructional strategies or new simulations. The user interface would support entering new medical content, instructional strategies and

student modeling. Initially the authoring tools would be usable only by programmers. Later development of interfaces for naive users, including instructors could be developed.

Application of the Authoring Tools

Authoring tools enable programmers and instructors to quickly and cost effectively develop, deliver and maintain intelligent NLP tutors for field application. These tools should reduce development costs and decrease the time and resources to maintain the tutor. Instructional developers and subject matter experts could use the tools to deliver instruction which would be 1) automatically generated while an instructor interacts with the tools and 2) delivered within a simulation and NLP dialogue environment.

The discourse modeling techniques designed were chosen in part to facilitate direct representation in a visual interface accessible to nonspecialists. In this section the design of authoring tools is illustrated by which layperson-authors can model various kinds of discourse. To illustrate different features and application domains, the examples include medical treatment of a conscious patient, and prisoner interrogation. It should be noted that the specifics of the interface design are preliminary, and could be refined according to usability testing. The major point is to illustrate how the underlying control structures described can be presented in a relatively intuitive manner.

Discourse is modeled at several levels of description. Therefore, authoring also takes place at different levels of description:

- One would diagram a "strategic network" to indicate the major phases of the discourse as it relates to the purposes of the application domain, and how the style of discourse might change between these phases.
- One would diagram a "tactical network" to indicate possible sequences of exchanges that meet particular discourse goals.
- One would writes individual "response rules," used as resources in the tactical networks to say how the simulated interlocutor might respond to particular utterances on the part of the user.
- One would defines "sentence categories," clusters of semantically related sentences that are treated the same, to provide the terms in which these response rules are written. Although the author would define these categories by providing particular natural language examples, once defined the underlying LCS is independent of these sentences, and could be applied to other languages.

At each level, one would construct a library of resources to be used and organized at the level above. The author has a choice of proceeding in a top-down manner, refining a strategic network into details; in a bottom-up manner, writing response rules first and then organizing them into tactical and strategic networks; or in a mixed approach. The approach taken to authoring could depend on personal preferences, or on the nature of the application. For example, in some applications there is a clear protocol for discourse on the part of the interlocutor being simulated, so it may be natural to begin with a description of this protocol as a strategic network. This is the case with the emergency medical examples given below. In other applications, the simulated interlocutor does not have a discourse agenda, but reacts to the utterances of the human user. In these cases it may be best to begin with a collection of possible responses to particular queries, as is done with the prisoner interrogation example below. Finally it is noted that it is entirely possible to provide automated guidance on how to use the following tools. For example, the computer can guide an author through top-down refinement of discourse resources.

Tutor development begins when the author encodes new nouns and verbs for the new domain. The system would support the creation of new dialogue management techniques as well, including pedagogical styles, dialogue models, exchange models, discourse goals, rhetorical acts and communicative acts. In addition to these dialogue features the author can describe additional pedagogical features of individual words. For instance, Figure 8 shows how the noun defibrillate might be used for tutoring, by listing how, when and within what procedure it might appear. The interactive dialogue mechanism so authored would be used in conjunction with a simulation based on knowledge units and built through a domain knowledge tool described above.

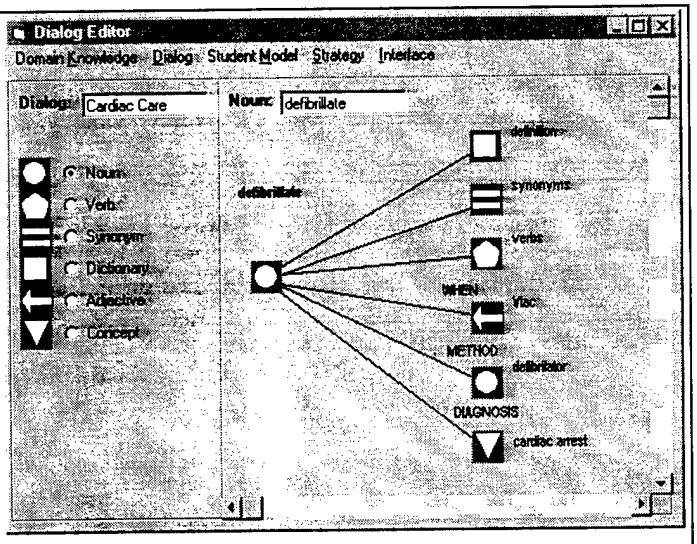


Figure 8. The dialogue editor

Authors would build tutoring systems by opening libraries of topics and augmenting them as needed. The topic list and domain model would be adjusted through the Domain Editor, Figure 9, which elicits additional information about facts, concepts and procedures and encodes pedagogical information about relationships between topics. The Domain Editor would be an interactive graphical tool that allows authors to design curriculum networks and presents relationships between domain concepts. Hierarchical links between topics indicate a variety of relationships, including component membership (e.g., subsumption), instructional goals (e.g., prerequisites), or concept network (e.g., typical, easy or difficult topics). For instance, in Figure 9 tension pneumothorax (a break in the chest cavity) is a subset of pneumothorax, which is a subset of chest wounds. Precursor topics are indicated, such that respiratory care (including airwaves, breathing and circulation) and cardiac care should be mastered before chest wounds are covered. Head wounds and limb wounds might be covered simultaneously.

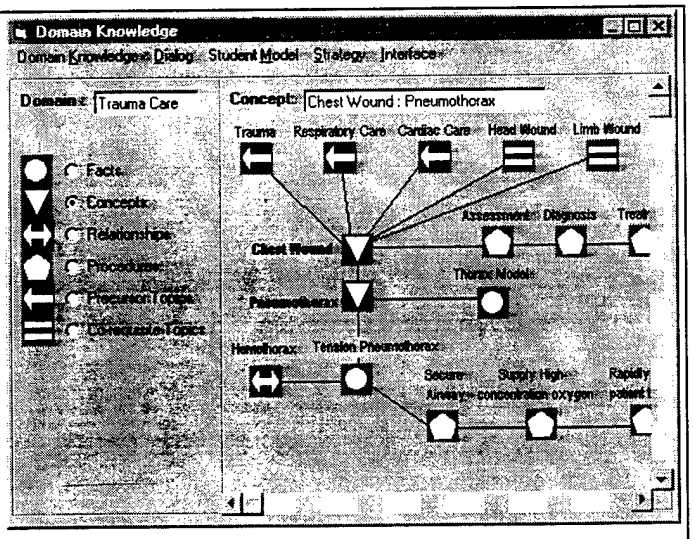


Figure 9. The domain editor

Lesson-building tools would exploit the information in the knowledge base and lexicon. For example, students can be taught the names of parts of the body because the author has given object names to parts of the body in the domain knowledge base. An instructor can record a procedure that a student must learn by inputting the steps of the procedure in the domain knowledge base. As shown in Figure 9, the steps needed for handling tension pneumothorax include 1) secure airway and 2) supply high-concentration oxygen. During training, the student would be prompted to carry out the correct sequence of actions and would receive meaningful generated feedback.

To begin, an example of authoring in a top-down fashion is presented. Suppose one wants to simulate the dialogue by which an emergency medical technician interacts with a patient, a dialogue that is somewhat stylized. Figure 10 illustrates one possible protocol for such discourse: establish contact with the patient (obtaining permission to treat if necessary); stabilize the patient (according to the "ABC-BS" protocol of Airway, Breathing, Circulation, Bleeding and Shock); and follow the "SOAP" protocol (subjective interview, objective examination, assess the patient's condition, and formulate a plan for treatment). The "*" arcs mean that the arc is to be taken for any response category not otherwise specified. Using a graphical interface such as that shown, a layperson-author who knows the protocol can easily construct a state transition network representing this protocol in diagrammatic form.

The diagram in itself is a complete representation of the structure of a genre of discourse at a

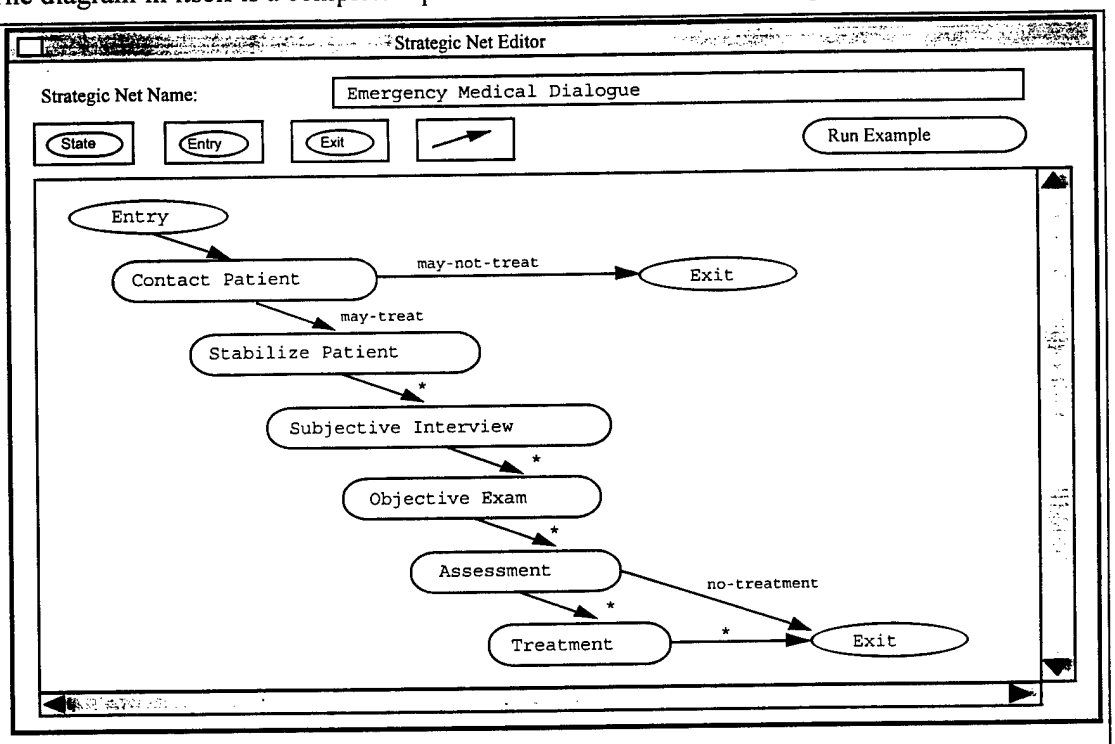


Figure 10. Strategic network editor

high level of abstraction. However it cannot be used until the details are specified. This is the subject of subsequent sections. For example, it can be shown how each strategic state can have its own interaction style, as expressed in tactical networks and response selection strategies. The authoring tools would help the author identify states with insufficiently specified components. These components could be accessed by hyperlinks from the strategic network, as well as directly by name. Thus, the strategic network also serves as a high level "map" to organize the work of authoring.

Once sufficient details have been worked out, the author could "run" the network to test it using the "Run Example" button. Each state would be highlighted as it is activated, while the author interacts in natural language within a separate window.

Instructional strategy tools would support a wide range of learning activities (e.g.. identification of topics components and relationship among operations, procedures, topics and concepts). The Tutoring Strategy Editor, Figure 11, will use a flow-line paradigm for graphically representing instructional decisions. Pedagogical exercises would be authored based on templates such as shown in Figure 11, which indicates how the system would monitor a student's ability to follow a procedure. For example, if the student makes an error the tutor would reason, based on paths input by the author, about whether to correct the missteps, suggest alternative steps, add missing steps, etc. It can also chose to reset or modify the simulation and to print a textual comment.

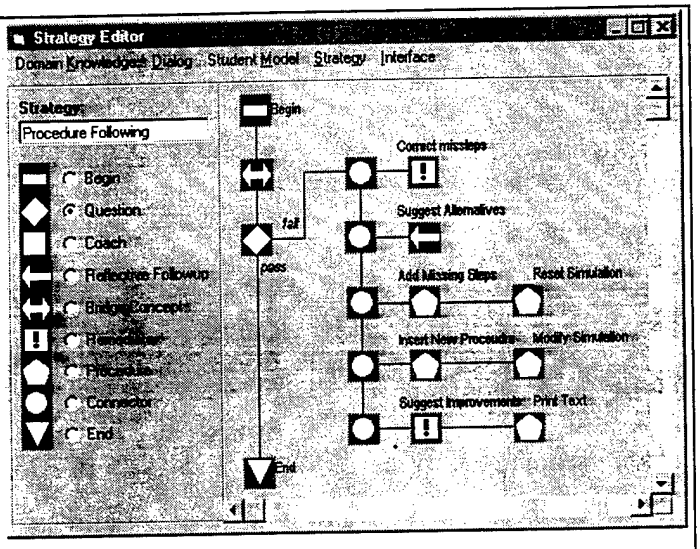


Figure 11. The tutoring strategy editor

The discussion now shifts to a bottom-up description of the designed authoring process. Suppose that one wanted to simulate interrogation of a prisoner of war. The simulation would be of the prisoner: the human user of the system plays the role of interrogator. In this case the prisoner does not control the dialogue, but rather reacts to the questions posed. A layperson-authorable way of specifying how a large set of questions that all share the same basic meaning can be responded to (by the system, simulating the prisoner) with members of another set of semantically related replies is needed.

The first step, before the author can specify response rules, is to enable this author to define the semantically related sets of utterances that the rules would recognize and produce. The LCS, discussed in this report, provides the kind of representation needed. However, authors who are not trained computational linguists cannot be expected to author LCS. The system would use machine learning techniques to enable the linguistic layperson to construct LCS without even realizing it, by giving example sentences.

Figure 12 illustrates an interface for this process. To define a "sentence category," the author enters several examples of sentences that have similar meaning and thus should belong in the category. The author type each sentence into the "example tester" field one by one. Optionally, the author can see if the sentence is already included in the current sentence category by clicking on "test this example."

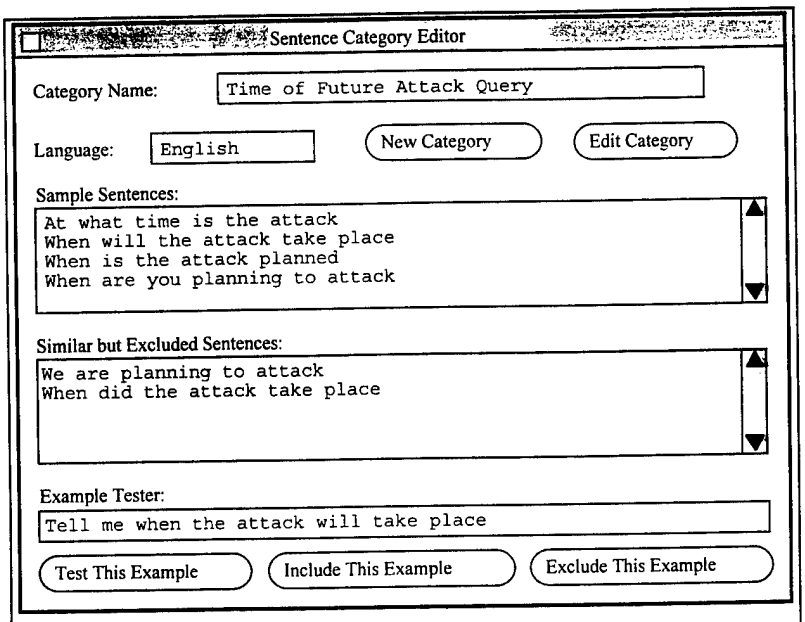


Figure 12. Sentence category editor

The software would reply with "included" or "not included," and perhaps further information concerning the nature of the mismatch if not included. Then, to extend the category to include the new sentence, the author clicks on "include this example." Behind the scenes, the software parses the sentence into LCS, and uses inductive generalization methods to modify the LCS pattern that represents the category to cover all of the positive example sentences, including the new one. If an example sentence that should not be accepted is accepted, this sentence can be marked as a negative example with the "exclude this example" button. In this case, the software uses version-space techniques from machine learning to restrict the LCS.

Through this process, the author can construct patterns representing a set of semantically related sentences, *including sentences other than those the author typed*, without needing to be aware of or understand the LCS representation and the machine learning techniques used. The first implementation of the sentence category interface would be for English. It would be possible to extend the tool to handle other languages, and to provide tools for porting sentence categories from one language to another.

Once the author has defined a collection of sentence categories, s/he writes "response rules" to indicate how to reply to a sentence category. Groups of response rules would then be used to write tactical states. Note that multiple response rules can be written for the same input sentence category. The response rule used for a given input sentence would be chosen according to the current tactical state.

Responses can be indicated at different levels of specificity. Sometimes it is sufficient to specify a fixed reply or set of semantically identical replies. Figure 13 shows a simple response rule that maps one sentence category to another, as indicated by the "use sentence category" selection of the reply type.

Having already defined sentence categories, the author simply indicates the sentence categories for inputs and replies. The "run example" button enables the author to type in an input sentence to which the rule would be applied.

The authoring interface would also provide the capability of indicating what kind of reply should be given when the content of the reply is not known in advance. The author would direct the dialogue system to plan a response based on information available in an application database or situation simulation. If the planning process results in more than one possible reply, the response selection strategy associated with the current strategic state would be used to select a response.

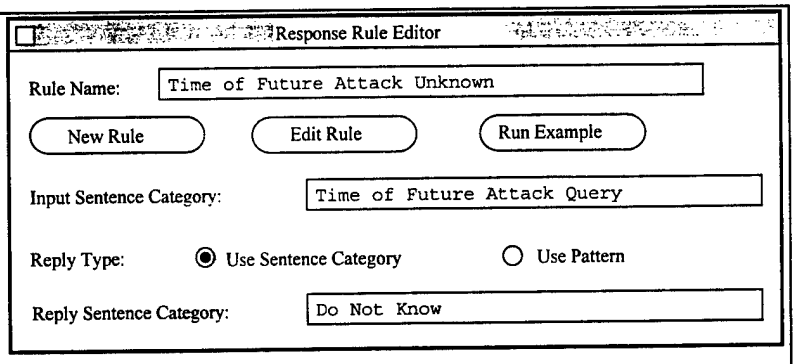


Figure 13. Response Rule editor: literal reply

Figure 14 shows the use of this authoring capability, which is selected by the "use pattern" button for the reply type. Using the three menus shown, the author has indicated that the system would respond to queries about the time of future attack with a description (the reply type) of the attribute value (the information type) of future attack times (the topics). The resulting pattern is displayed in the "reply pattern" field, and can be directly edited if desired. Diagrammatic displays of term hierarchies would be provided in addition to menus to help the author navigate the options for "reply type," "information type," and "topics." The "reply type" and "information type" options would be pre-defined and portable across domains. The "topics" would be authored for each domain, using the sense editor described

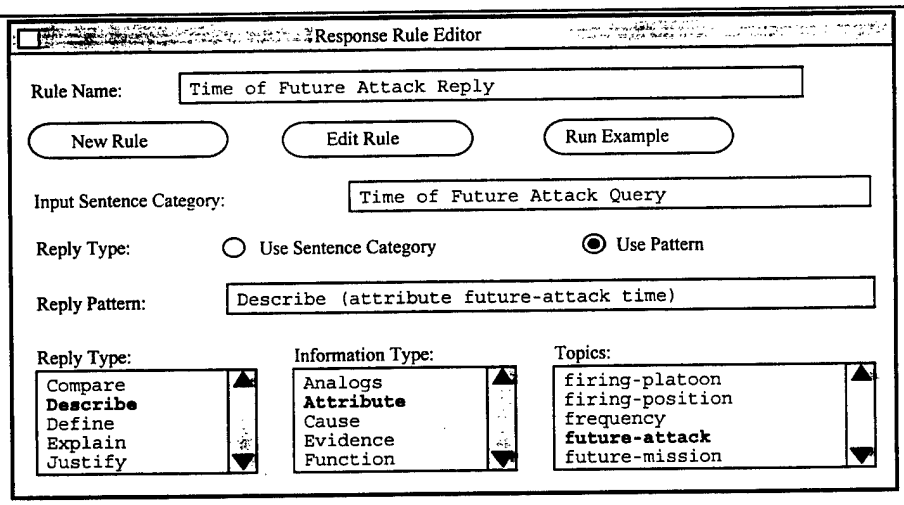


Figure 14. Response Rule editor: planned reply

earlier.

The completed tutor would contain a list of learning objectives that must be realized by the student. As the student progresses through the simulation and through the dialogue with the tutor, the system would maintain a model of the student's knowledge, based on course objectives. Decisions about which topics to present next, or how to bias the curriculum would be controlled by the relationship among course objectives and the state of the student model.

The Student Model Editor would allow instructors to specify how student behavior would be used to infer student knowledge states and mastery level, see Figure 15. The tutor would track the student's proficiency on all topics.

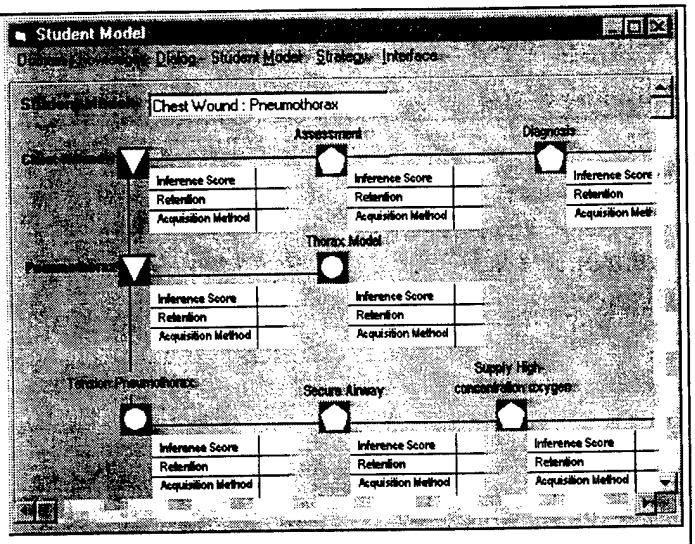


Figure 15. The student model editor

This numerical proficiency rating would be a history of the student's performance and would be stored along with the student's acquisition factor, i.e., how quickly the student learns new information. Data is gathered by examining the response of the student to see how quickly the student learns each procedure. The retention factor would measure how well the student recalls the material over time. This number is calculated by examining how well the student performs at skills that have not been presented for a certain length of time. These general factors should be predictive of overall learning and would allow for a more accurate response by the tutor to the idiosyncrasies of the student.

The authoring tools would provide template screens for use by instructors. The Interface Editor, Figure 16, would allow authors to construct student screens and to include simulations, videos, or multimedia environments. A pallet would allow for the creation and inclusion of categories of screen features, including sliders, movies, tables, hot spots, dialogues, and text. For instance, a generic template for simulation interaction, multiple choice questions or video display would be selectable by the instructor, who can create a number of presentation contents to fill the templates with specific values, i.e. a simulation or text.

Evaluation of the tools would be made by asking programmers and instructional designers to make derivatives of the NLP medical system. These derivatives would account for most of the tutoring strategies for teaching in the medical domain. Several examples of dialogues generated by the system would be displayed to demonstrate the generality the knowledge base. It can be demonstrated that the knowledge base contains generic instructional strategies for teaching in the medical domain and this could be tested empirically with a number of users and domains. It is expected that the tools designed for this system will be highly usable without sacrificing tutoring systems power.

An infrastructure for representing planning and discourse behavior as described in this section would be designed, and then the resulting discourse would be analyzed to measure how it can simulate a variety of human tutorial interactions. The computer produced dialogue would be compared to

documented instructional strategies used by instructional designers. The long term goal would be to demonstrate that the resulting system can simulate a variety of human tutorial interactions.

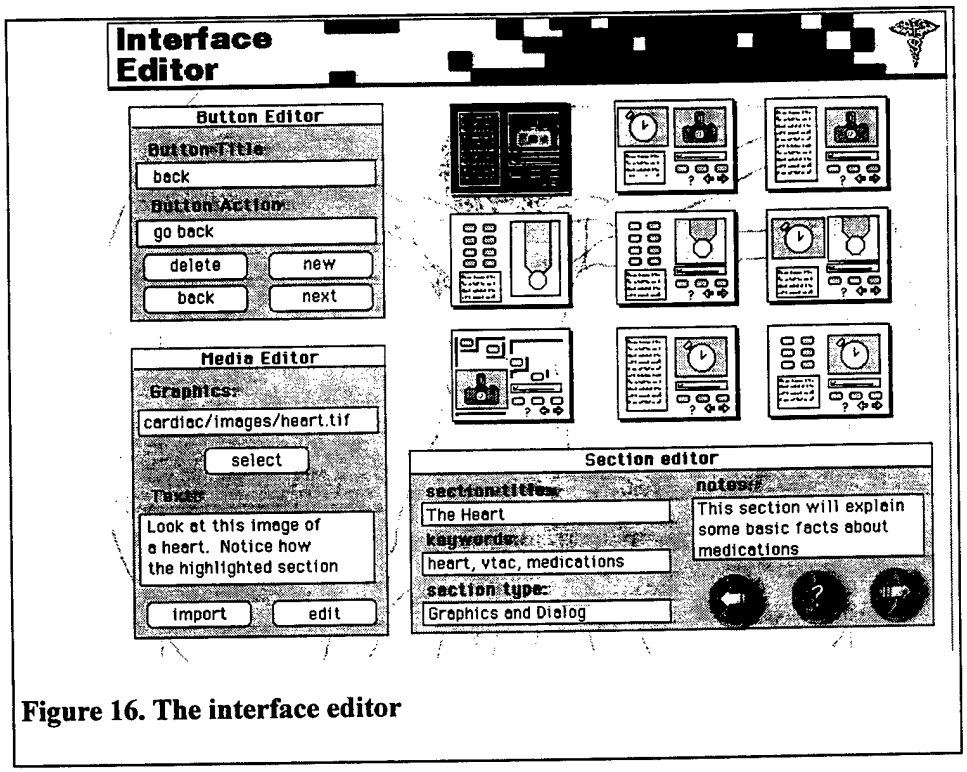


Figure 16. The interface editor

To summarize the foregoing discussion, Figure 17 illustrates the conceptual relationships between the authorable components of the hierarchical discourse model. Using an intuitive graphical interface, an author would diagram a strategic network to organize the major components of a discourse model. The diagram would also serve to organize the work of authoring these components. States of a strategic network provide access to response selection strategies, to be applied at the level of response rules if needed, and access to tactical networks. The author diagrams tactical networks to indicate how the manner in which the system behaves changes as one moves through the dialogue. States of the tactical network are easily defined by selecting from lists of response rules, and by inventing response categories used to control movement through the networks. Individual response rules are authored by selecting from lists of sentence categories, and by selecting combinations of rely types, information types, and topics. Finally, sentence categories are authored in an intuitive manner, by providing positive and negative examples. The specifics of the foregoing design are meant to show feasibility of authoring by laypersons, and would be subject to change during implementation and usability testing.

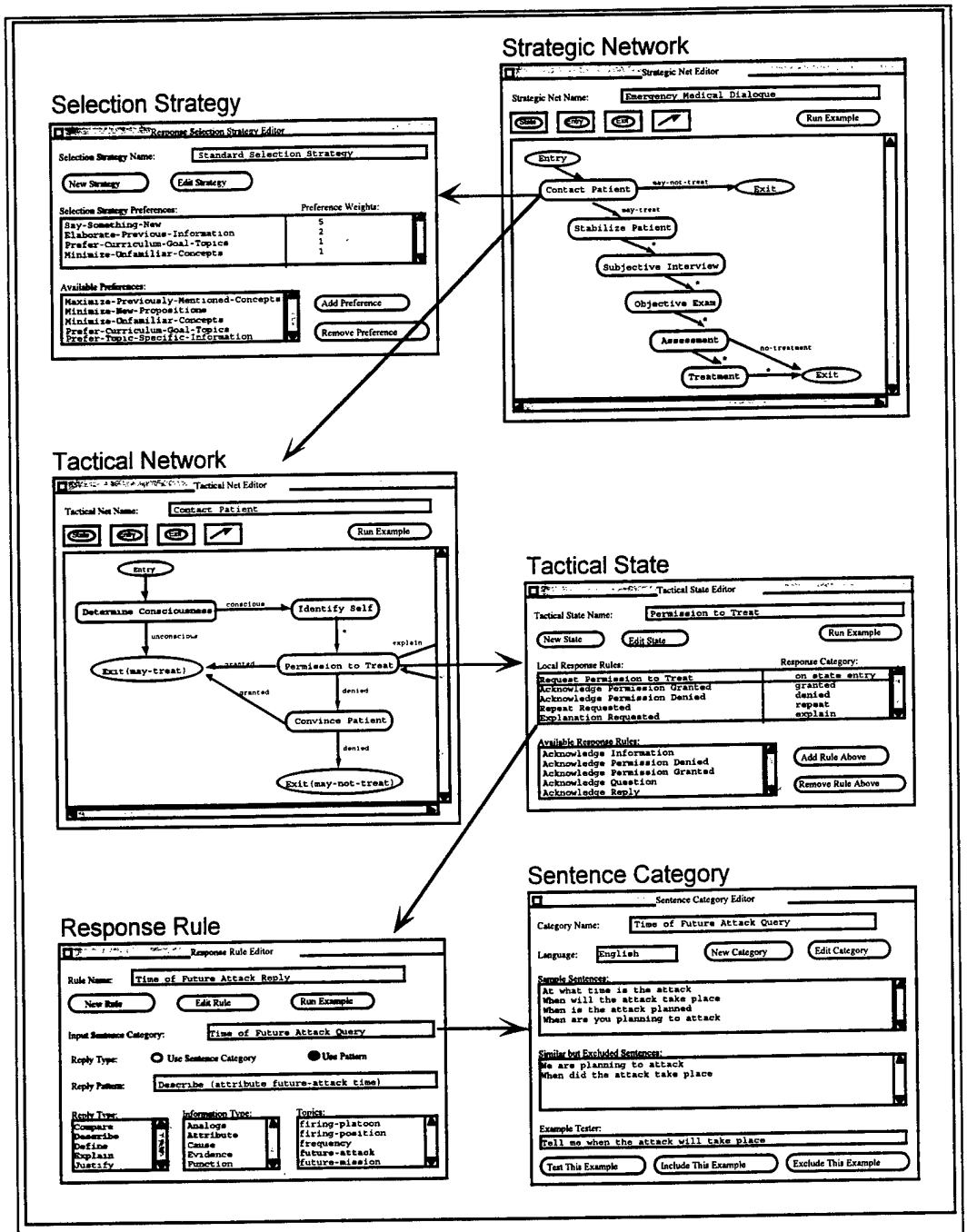


Figure 17. Summary of relations between authorable resources

Task 5: Integrate components

All components including the knowledge base, English NLP, and LCS Authoring system would need to be integrated with the revised MILT code. This step would consist of the determination of the algorithms and data exchange protocols that would be necessary for the various components to function as a complete system.

Task 6: Expand Internet Capabilities

During Phase I, a small web page for MILT (http://www.maad.com/MILT) was constructed. The constructed page illustrates many of the features that would be useful for users of MILT, but could easily be expanded to meet additional needs.

Task 7: Generate Military Dialogue Based Lesson

If the system is developed, a lesson should be designed to provide a demonstration of each of the dialogue capabilities – including dialogue and LCS authoring. During Phase I, emergency medical treatment was chosen as the starting application domain. This domain was chosen for its ability to be expanded to other similar domains and for its applicability to both Government (the Army in particular) and commercial training needs.

Summary

This report documented the specifications of a authorable dialogue-based military information and procedural tutor. It focused on the concept for development of an authorable dialogue system capable of supporting military and procedural informational needs. Major components of the designed system are a English natural language processing (NLP) system, a artificially intelligent knowledge base, and a lexical conceptual semantic (LCS) editor. This concept is based upon the successful design and proof of concept analysis that was performed in a Phase I Small Business Innovative Research (SBIR) effort. That effort determined that it is feasible to link a English NLP system to a AI dialogue system to develop a fully integrated dialogue-based tutoring system.

References

Abney, S. (1989) A Computational Model of Human Parsing, Journal of Psycholinguistic Research, 18, (1).

Abney, S. (1986) Licensing and Parsing, in Proceeding of NELS XVII.

Altmann, G. (1988) Ambiguity, Parsing Strategies, and Computational Models, Language and Cognitive Processes, 3(2) , 73-98.

Bach, K. & Harnish, R. M. (1979). Linguistic Communication and Speech Acts. Cambridge: MIT Press.

Baltin (1989) Heads and Projections, in Alternative Conceptions of Phrase Structure (Baltin and Kroch, eds). Chicago, IL: University of Chicago Press.

Barton, E. and R. Berwick (1985) Parsing with Assertion Sets and Information Monotonicity, in Proceeding of the International Joint Conference on Artificial Intelligence., 669-671.

Berwick, R. (1985) The Acquisition of Syntactic Knowledge. Cambridge, MA: MIT Press.

Berwick, R. and A. Weinberg (1984) The Grammatical Basic of Linguistic Performance. Cambridge, MA: MIT Press.

Bever, T. (1968) A Survey of Some Recent Work in Psycholinguistics. in Specification and Utilization of a Transformational Grammar (W. Plath, ed). Scientific report number 3. New Jersey: T. J. Watson Research Center, IBM Corporation.

Carlson, G. and M. K. Tanenhaus (1988) Thematic Roles and Language Comprehension, in Syntax and Semantics 21: Thematic Relations. New York: Academic Press.

Chomsky, N. (1992) A Minimalist Program for Linguistic Theory. MIT Occasional Papers in Linguistics, 1. Cambridge, MA:MIT Working Papers in Linguistics.

Chomsky, N. (1986a) Barriers. Cambridge, MA: MIT Press.

Chomsky, N. (1986b) Knowledge of Language: Its Nature, Origin, and Use. New York: Praeger.

Chomsky, N. (1981) Lectures on Government and Binding. Dordrecht: Foris Publications.

Elhadad, M. (1992). Using Argumentation to Control Lexical Choice: A Functional Unification-Based Approach. PhD thesis, Computer Science Department, Columbia University.

Eliot, C. and Woolf, B. O., Iterative Development and Validation of a Computer-based Medical Tutor, in Intelligent Tutorial Systems, ITS'96, Lecture notes in Computer Science 1086, C. Frasson, G. Gauthier and A. Lesgold (EDs.), Springer: Berlin, 1996a.

Eliot, C.R. and Woolf, B. P., A simulation-based Tutor that reasons about Multiple Agents, Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96), Portland, OR. pp 409-415, 1996b.

Ferreira, F. and C. Clifton (1986) The Independence of Syntactic Processing, Journal of Memory and Language, 25, 348-368.

Fodor, J. A. (1983) The Modularity of the Mind. Cambridge, MA: MIT Press.

Ford, M., J. Bresnan, and R. Kaplan (1982) A Competence-based Theory of Syntactic Closure, in The Mental Representation of Grammatical Relations (J. Bresnan ed.). Cambridge, MA: MIT Press.

Forster, K (1979) Levels of Processing and the Structure of the Language Processor, in Sentence Processing: Studies Presented to Merrill Garrent (W. E. Cooper and E. C. T. Walker, eds). Hillsdale, NJ: Lawrence Earlbaum Associates.

Frazier, L. and J. Fodor (1978) The Sausage Machine: A New Two- stage parsing model. Cognition 6,291-325.

Frazier, L. and K. Rayner (1982) Making and Correcting Errors during Sentence Comprehension: Eye Movement in the Evaluation of Structurally Ambiguous Sentences. Cognitive Psychology 14, 178-210.

Frazier, L. (1987) Syntactic Processing: Evidence from Dutch, Natural Language and Linguistic Theory 5, 519-559.

Fukui, N. and M. Speas (1986) Specifiers and Projections. In MIT Working Papers in Linguistics 8, Cambridge.

Garman, J. (1997) A Computational Model of Human Language Processing and Its Uses in Everyday Life. Ph.D. dissertation, University of Maryland, College Park, MD.

Gibson, E. (1991) A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA.

Grosz, B. & Sidner, (1985). Discourse structure and the proper treatment of interruptions. Proc 9th IJCAI. Los Angeles, CA. pp. 832-839.

Gorrell, P. (1994a) Syntax and Perception. Cambridge University Press. (forthcoming)

Gorrell, P. (1994b) Minimal Commitment Theory: Some Clarifications. MS, University of Maryland, College Park, MD.

Gorrell, P. (1992) The Parser as Tree Builder. MS, University of Maryland, College Park, MD.

Gorrell, P. (1987) Studies in Human Syntactic Processing: Ranked- Parallel versus Serial Models. Ph.D. dissertation, University of Connecticut, Storrs, CT.

Hovy, E. (1990). Pragmantics and natural language generation. Artificial Intelligence. 43(1), 153-197.

Katz, S., & Lesgold, A. (1993). The role of the tutor in computer-based collaborative learning situations. In S. Lajoie & S. Derry (Eds.), Computers as cognitive tools ,289-317. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kayne, R. (1984) Connectedness and Binary Branching. Dordrecht: Foris Publications.

Kimball, J. (1973) Seven Principals of Surface Structure Parsing in Natural Language. Cognition 2, 15-47.

Marcus, M., D. Hindle and M. Fleck (1983) Talking about Talking about Trees, in Proceeding of the 21st Annual Meeting of the Association for Computational Linguistics, 129-136.

Marcus, M. (1980) A Theory of Syntactic Recognition for Natural Language. Cambridge, MA: MIT Press.

Marslen-Wilson, W. and L. Tyler (1987) Against Modularity, in Modularity in Knowledge Representation and Natural Language Understanding (J. Garfield, editor). Cambridge, MA: MIT Press.

Merlo, P. (1992) On Modularity and Compilation in a Government- Binding Parser. Computer Science Technical Report Series: CS-TR- 2982. University of Maryland, College Park, MD.

Mitchell, T. M. (1977). Version spaces: A candidate elimination approach to rule learning. Proc. 5th IJCAI, Cambridge MA, 305-310.

Moore, J. D. (1995). Participating in Explanatory Dialogues: Interpreting and Responding to Questions in Context. Cambridge: MIT Press.

Moore, J. D. & Paris, C. L. (1993). Planning Text for Advisory dialogues: Capturing intentional and rhetorical information. Computational Linguistics 19(4), 651-695.

Munn, A. (1993) Topics in the Syntax and Semantics of Coordinate Structures, Doctoral Disseration, University of Maryland, College Park, MD.

Murray, T. (1996a). Having it all, Maybe: Design Tradeoffs in ITS Authoring Tools, In Proc. of ITS 96, Third international Conf. on Intelligent Tutoring Systems. Berlin: Springer-Verlag.

Murray, T. (in press). Authoring Instructional Expertise in Knowledge based Tutors, Instructional Science, Kluwer Academic Publishers.

Murray, T & Woolf, B., (1991) A knowledge Acquisition Framework for Intelligent learning Environments. ACM SIGART bulletin, 2(2), 1-13.

Pritchett, B. (1992) Grammatical Competence and Parsing Performance. Chicago, IL: University of Chicago Press.

Pritchett, B. (1988) Garden Path Phenomena and the Grammatical Basis of Language Processing, Language 64, 539-576.

Stowe, L. (1989) Thematic structure and Sentence Comprehension, in Linguistic Structure in Language Processing (G. N. Carlson and M. Tanenhaus, eds). Kluwer Academic Publishers: Dordrecht.

Stowell, T. (1981) Origins of Phrase Structure. Ph.D. dissertations, MIT, Cambridge, MA.

Swinney, D. (1979) Lexical Access during Sentence Comprehension: (Re)consideration of Context Effects, Journal of Verbal Learning and Verbal Behavior, 18, 645-659.

Suthers, D. (1993a). Preferences for model selection in explanation. Proc.. 13th IJCAI.

Suthers, D. (1993b). An Analysis of Explanation and its Implications for the Design of Explanation Planners. Ph.D. Dissertation, University of Massachusetts, Amherst.

Suthers, D. (1991). A task-appropriate hybrid architecture for explanation. Computational Intelligence, 7, 315-333.

Suthers, D. & Jones, D. (1997). An Architecture for Intelligent Collaborative Educational Systems. To appear in AI&ED '97, Kobe Japan.

Tanenhaus, M. K., J. M. Leiman and M. S. Seidenberg (1979) Evidence for Multiple Stages in the Processing of Ambiguous Words in Syntactic Contexts, Journal of Verbal Learning and Verbal Behavior, 18, 427-440.

Weinberg, A. (1993a) A Parsing Theory for the Nineties: Minimal Commitment, Language and Cognitive Processes, to appear.

Weinberg, A. (1993b) Parameters in the Theory of SentenceProcessing: Minimal Commitment Goes East, Journal of Psycholinguistic Research 22, 339-364.

Weizenbaum, J. 1966. ELIZA - A computer program for the study of natural language communication between man and machine. *CACM 9:36-45.*

Woolf, B. & McDonald, D. (1984). Building a computer tutor: Design issues. *IEEE Computer.* September 1984, 61-73.

Wright, B. and M. Garrett (1984) Lexical Decision in Sentences: Effects of Syntactic Structure, Memory and Cognition, 12, 31-45.