

UNIVERSITY OF CALIFORNIA

Los Angeles

Computer Generation of Meta-Technical Utterances  
in Tutoring Mathematics

A dissertation submitted in partial satisfaction of the  
requirements for the degree of Doctor of Philosophy  
in Computer Science

by

Ingrid Zukerman

1986

© Copyright by  
Ingrid Zukerman  
1986

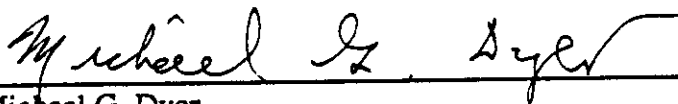
The dissertation of Ingrid Zukerman is approved.



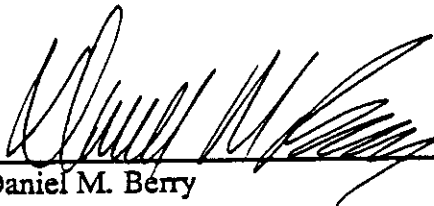
Richard J. Shavelson



Kirby A. Baker



Michael G. Dyer



Daniel M. Berry



Judea Pearl, Committee Chair

University of California, Los Angeles

1986



## DEDICATION

To Moshe, Jenny, Debbie, Ashley and Wendy.



## TABLE OF CONTENTS

	page
1 Introduction and Overview .....	1
1.1 Why Algebra? .....	4
1.2 Input and Output of FIGMENT .....	5
1.3 Text Generation Techniques .....	6
1.4 Tutoring Systems .....	10
1.4.1 Intelligent Tutoring Systems .....	11
1.5 Overview of this Dissertation .....	15
2 Semantics of Meta-Technical Utterances .....	17
2.1 Sample Expressions Used in Tutoring Algebra .....	17
2.2 Overview of Related Research .....	19
2.3 A Functional Taxonomy for the Generation of MTUs .....	26
2.3.1 Knowledge-Organization MTUs .....	26
2.3.2 Knowledge-Acquisition MTUs .....	29
2.3.3 Affect-Maintenance MTUs .....	33
2.3.4 Summary of Meta-Technical Utterances .....	35
3 Design of FIGMENT .....	37
3.1 Problem Solving Expertise Module .....	38
3.1.1 Domain Knowledge .....	39
3.1.2 Problem Solving Knowledge .....	45
3.1.3 Output of the Problem Solving Expertise Module .....	46
3.2 Model of the Knowledge of a Student .....	49
3.3 Tutoring Strategies Module .....	52
3.3.1 Output of the Tutoring Strategies Module .....	54
4 Generation of Knowledge-Acquisition MTUs: The Comprehension-Processes Module .....	57
4.1 Determining Focus .....	60
4.1.1 Recognizing a Permanent Focus Shift .....	63
4.1.2 Recognizing a Temporary Focus Shift .....	69
4.2 Determining Type of Relationship .....	72
4.3 Selecting Implementation Mode .....	75
4.4 Applying Mental Resources .....	83
4.4.1 Preparing Computational Power .....	84
4.4.2 Preparing Processing Time .....	89
4.5 Building up Motivation and Justification .....	96
4.5.1 Selecting a Motivation .....	106
4.6 Partial Output of the Comprehension-Processes Module .....	110
5 Generation of Knowledge-Organization MTUs .....	115
5.1 Generation of Additive MTUs .....	115
5.2 Generation of Adversative MTUs .....	117
5.3 Generation of Causal MTUs .....	127
5.4 Generation of Attributive MTUs .....	130

5.5	Generation of Temporal MTUs .....	131
5.6	Output of the Comprehension-Processes Module .....	134
6	Generation of an English-Language Representation: The Sentence Composer .....	140
6.1	The Phrasal Dictionary .....	141
6.1.1	Generating an English Representation of a Dictionary Entry .....	143
6.1.2	Updating the Dictionary .....	145
6.1.3	Generating Pronouns .....	146
6.2	The Attribute-Clause Generator .....	147
6.2.1	Some Heuristics for the Generation of Attribute Clauses .....	151
6.3	The Utterance Generators .....	155
6.3.1	Structure of the Generated Text .....	157
6.3.2	Selection of an English Representation .....	160
6.3.3	The Declarative Generator .....	160
6.3.3.1	The Topic-Introduction Generator .....	162
6.3.3.2	The Equation-Explanation Generator .....	167
6.3.4	The Procedural Generator .....	172
6.3.4.1	The Alternative Generator .....	173
6.3.4.2	The Rule-Sequence Generator .....	177
6.3.4.3	The Rule-Cluster Generator .....	179
6.3.4.4	The Rule Generator .....	184
6.3.4.5	The Pattern Generator .....	185
6.3.4.6	The Expectation Generator .....	186
6.3.4.7	The Result Generator .....	187
6.3.4.8	The Finish Generator .....	188
6.3.4.9	The Continue Generator .....	188
6.3.5	The Intervening-Utterances Generator .....	189
6.3.5.1	The Method Generator .....	189
7	Conclusions and Future Research .....	192
7.1	Limitations of this Research and Future Work .....	192
	Appendix 1 Taxonomy of Conjunctions by Wilkins .....	195
	Appendix 2 Output of the Tutoring Strategist .....	198
	Appendix 3 Arguments of the Technical Part of a Technical Utterance .....	211
3.1	Arguments of the Technical Part of a Rule Technical Utterance .....	211
3.2	Arguments of the Technical Part of a Pattern Technical Utterance .....	212
3.3	Arguments of the Technical Part of an Expectation Technical Utterance .....	213
	Appendix 4 Referencing Mathematical Entities .....	214
	Appendix 5 Output of the Comprehension-Processes Module .....	216
5.1	Topic Extended-Message .....	216
5.2	Equation Extended-Message .....	217
5.3	Alternatives Extended-Message .....	218
5.3	Alternative Extended-Message .....	218
5.4	Rule Extended-Message .....	219



5.5 Pattern Extended-Message .....	220
5.6 Expectation Extended-Message .....	221
5.7 Result Extended-Message .....	221
5.8 Finish Extended-Message .....	222
5.9 Continue Extended-Message .....	222
5.10 Method Extended-Message .....	222
5.11 Report Extended-Message .....	223
Appendix 6 Thresholds for the Generation of Estimational MTUs .....	224
6.1 Thresholds for Complexity-Related MTUs .....	224
6.2 Thresholds for Length-Related MTUs .....	226
Appendix 7 Rules Applied for Selecting a Motivation .....	229
Appendix 8 Typical Output of the Comprehension-Processes Module and Sentence Composer .....	236
8.1 A Linear Equation .....	236
8.2 A Quadratic Equation .....	240
8.3 A Third Degree Equation .....	246
REFERENCES .....	251



## LIST OF FIGURES

	page
Fig. 1.1: Text with MTUs .....	2
Fig. 1.2: Text without MTUs .....	2
Fig. 1.3: Output of the Tutoring Strategist .....	6
Fig. 1.4: Block diagram of an Intelligent Tutoring System .....	12
Fig. 2.1: Summary Table of Conjunctive Relations by Hallyday and Hasan ...	24
Fig. 2.2: Summary of Meta-Technical Utterances .....	36
Fig. 3.1: Block Diagram of FIGMENT .....	39
Fig. 3.2: Sample Rule in Problem Solving Expertise Module .....	42
Fig. 3.3: Problem Solving Knowledge of the PSE Module .....	46
Fig. 3.4: Typical Output of the Problem Solving Expertise Module .....	48
Fig. 3.5: Sample Rule in Model of the Knowledge of a Student .....	51
Fig. 3.6: Technical Part of Sample Output of the Tutoring Strategist .....	56
Fig. 4.1a: Process for Generating a Permanent Focus-shift MTU for a Topic	65
Fig. 4.1b: Process for Generating a Permanent Focus-shift MTU for an Equation .....	66
Fig. 4.1c: Process for Generating a Permanent Focus-shift MTU for an Alternative .....	67
Fig. 4.1d: Process for Generating a Permanent Focus-shift MTU for an Algebraic Rule .....	69
Fig. 4.2: Process for Generating a Temporary Focus-shift MTU .....	73
Fig. 4.3a: Process for Generating an Implementational MTU .....	76
Fig. 4.3b: Process for Generating an Implementational MTU for an Algebraic Rule .....	81
Fig. 4.4: Process for Generating a Complexity-related MTU .....	86
Fig. 4.5a: Process for Generating a Length-related MTU .....	92

Fig. 4.5b: Process for Generating a Length-related MTU for a Previous Topic-length Prediction .....	95
Fig. 4.6: Motivation Relations in the Context Hierarchy .....	100
Fig. 4.7: Process for Generating a Motivational MTU .....	102
Fig. 4.8: Sample Input to Comprehension-Processes Module .....	111
Fig. 4.9: Requirement-codes for Knowledge-Acquisition MTUs for Sample Input .....	113
Fig. 5.1: Sample Input to Comprehension-Processes Module .....	136
Fig. 5.2: Completed MTU Requirement-codes for Sample Input .....	139
Fig. 6.1: Segment of the Phrasal Dictionary .....	145
Fig. 6.2: Hierarchy of Generators in the Sentence Composer .....	157
Fig. 2app.1: Input Sequence of Technical Utterances .....	200
Fig. 6app.1: Selected Values of Talent, Relative-Expertise and Complexity .....	226
Fig. 6app.2: Thresholds Used to Generate Length-related MTUs .....	227
Fig. 6app.3: Some Values of Diligence and Length .....	228

## ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my advisor, Judea Pearl, for his insightful comments, the many hours of discussion he devoted to this research in its formative stages, and for his guidance and encouragement through all the stages of this dissertation.

Sincere thanks are extended to the dissertation committee members, Richard J. Shavelson, Kirby A. Baker, Michael G. Dyer and Daniel M. Berry for their useful comments. In particular, I would like to acknowledge Daniel M. Berry's stylistic suggestions and his advice in formatting and editing this document.

I am also indebted to Leonard Kleinrock, who generously shared with me his computational facilities supported under grant MDA-903-82-C-0064. It was due to these facilities that the process of programming this system was as painless as possible. Lillian Larijani deserves special thanks for going out of her way to help in various administrative matters; and I thank Tovah Hollander and Lance Lee for performing stylistic improvements to parts of this dissertation.

An enormous THANKS goes to my husband, Moshe, for his unwavering support and understanding throughout my studies, and for helping in every possible way during these last hectic months. Last but not least, I thank my children, Jenny, Debbie, Ashley and Wendy, for being so cute.

This research was supported in part by the National Science Foundation grants  
IST 81-19045 and DCR 83-13875.

## VITA

February 25, 1955	Born, Lima, Peru
1976	B.Sc., Technion - Israel Institute of Technology
1976-1977	Teaching Assistant, Technion - Israel Institute of Technology
1977-1979	Research Assistant, IBM Israel Scientific Center
1979	M.Sc., Technion - Israel Institute of Technology
1982-1986	Research Assistant, School of Engineering University of California, Los Angeles

## PUBLICATIONS AND PRESENTATIONS

- Zukerman I. (1979), *Development of a Data Base for Agricultural Planning*. Master Thesis, Technion - Israel Institute of Technology, Haifa, Israel, August 1979.
- Zukerman I. and Rodeh M. (1979), *Compaction in a Collection of Buddy Systems*. Technical Report 074, IBM Israel Scientific Center, September 1979.
- Zukerman I. (1981), *The Subsumption Problem for Relational Databases*. *Quarterly*, Computer Science Department, University of California, Los Angeles, Spring 1981.
- Zukerman I. and Pearl J. (1984), *Listener Model for the Generation of Meta-Technical Utterances in Math Tutoring*. Technical Report CSD-840064, Computer Science Department, University of California, Los Angeles. Also presented at the Conference of the Southern California Artificial Intelligence Society, October 1984.
- Zukerman I. (1985), *A Functional Taxonomy of Meta-Content Utterances*. In *Proceedings of the First Annual Artificial Intelligence & Advanced Computer Technology Conference*, pp. 1-7, May 1985.
- Zukerman I. and Pearl J. (1986), *Comprehension-Driven Generation of Meta-Technical Utterances in Math Tutoring*. To appear in *AAAI-86 Proceedings*, August 1986.





## ABSTRACT OF THE DISSERTATION

### Computer Generation of Meta-Technical Utterances in Tutoring Mathematics

by

Ingrid Zukerman

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1986

Professor Judea Pearl, Chair

A technical discussion often contains conversational expressions like: “however,” “as I have stated before,” “next,” etc. These expressions, denoted *Meta-Technical Utterances* (MTUs), carry important information which the listener uses to speed up the comprehension process. The goal of this research is to understand the semantics of text containing MTUs, the mechanisms by which people generate them, and the processes required for generating them mechanically. To achieve this goal, we model the meaning of MTUs in terms of their anticipated effect on the listener comprehension, and use these predictions to select MTUs and embed them in a computer generated discourse. This paradigm was implemented in a computer system called FIGMENT, which generates commentaries on the solution of algebraic equations.

We classify MTUs according to their function, as seen by the speaker, in transmitting the subject matter to the listener, and distinguish among three main types

of MTUs: (1) Knowledge Organization, (2) Knowledge Acquisition, and (3) Affect Maintenance. Knowledge-Organization MTUs reflect the organization of the material in the speaker's mind (e.g., "however," "in order to"), Knowledge-Acquisition MTUs provide information that enables the listener to prepare adequate knowledge-assimilating facilities (e.g., "we shall now introduce," "as I have stated before"), and (3) Affect-Maintenance MTUs convey the affective impact of an event (e.g., "fortunately"), and foster a positive attitude in the listener (e.g., "I shall go over this explanation again").

This classification governs the generation of MTUs in the following manner: Knowledge-Organization and some Affect-Maintenance MTUs are generated directly from the organization of the system's knowledge of the subject matter; Knowledge-Acquisition MTUs and the majority of Affect-Maintenance MTUs are generated by consulting simplified models of some mental processes which the user presumably activates upon encountering a technical message. For example, determining the context in which a technical message should be processed, building up motivation to attend to the next item of discourse, and so on.

The main contribution of this dissertation is the presentation of an explicit model for the generation of MTUs. This model can be incorporated into a text-generation facility to enable the generation of fluent and coherent discourse.

## CHAPTER 1

### Introduction and Overview

In the process of generating discourse, speakers and writers must decide what to say and how to present the information effectively. A speaker performs the first task by selecting appropriate items from his body of knowledge and ordering them into a coherent unit. In order to perform the second task, he determines which elocutionary acts to perform, such as asking a question or giving an explanation. He then decides which words to use, and how to group them into sentences.

Often we find not only the bare facts in texts, but also expressions like “however,” “as I have stated before,” “next” or “generally speaking.” It is our contention, that these expressions, denoted *Meta-Technical Utterances* (MTUs), are not included in the text merely for decorative purposes, but carry information which has a significant impact on making the subject under consideration better understood. Therefore, they are especially useful in situations where the listener is not familiar with the information presented by the speaker, as in a tutorial setting.

Our claim of the importance of MTUs is supported by Farnes (1973), Winter (1968 and 1977), Reichman (1978 and 1984) and Hoey (1979), who conclude that the presence of MTUs can *signpost* what kind of information is to be presented in the following sentence or sentences.

Let us illustrate the importance of these expressions by means of two commentaries presented in Figs. 1.1 and 1.2, on the same algebraic equation. Both contain the same information items; however, only the first commentary contains MTUs.

Now, let's consider a third degree example:

$$x^3 - x^2 - x + 1 = 0$$

There is a formula for solving third degree equations, but we haven't covered it in class. So we will have to try another approach. The most useful approach is to try to apply the factor theorem, so that we can reduce the problem to the solution of a linear and a quadratic equation.

*Exemplify factoring*

⋮

**Fig. 1.1: Text with MTUs**

$$x^3 - x^2 - x + 1 = 0$$

There is a formula for solving third degree equations. Apply the factor theorem. Reduce the problem to the solution of a linear and a quadratic equation.

*Exemplify factoring*

⋮

**Fig. 1.2: Text without MTUs**

A student who is not familiar with the subject of third degree equations could completely misunderstand the text without meta-technical utterances. He might think that applying the factor theorem is part of the formula for solving this type of equa-

tion, and that the next operation consists of reducing the problem to the solution of a linear and a quadratic equation.

The presence of MTUs has a special significance in a machine generated text, beyond speeding up the user's comprehension process. Since the user receives a tangible and frequent reinforcement of the machine's knowledge about the state of the discourse, he is provided with a perception of an intelligent interlocutor, thereby increasing the trustworthiness of the system.

Thus, the first goal of this research is to understand the semantics of texts containing MTUs, including the mechanisms by which people generate them and the processes required for generating them mechanically. To achieve this goal, we model the meaning of meta-technical utterances in terms of their anticipated effect on a listener's comprehension.

The prevailing paradigm in Artificial Intelligence is that adequate models of mental processes should be expressed by computer programs and evaluated by their performance, i.e., a computer program reflecting a model should be able to approximate human behaviour. Therefore, a second goal in this research is to weave MTUs into machine-generated discourse in order to achieve a natural effect. As a prerequisite for the attainment of this goal, we need to find out what kind of knowledge should be incorporated into a language-generation system.

We have selected the domain of high-school algebra as a testbed for our ideas, and have implemented some components of an Intelligent Tutoring System called FIGMENT to generate commentaries on the solution of algebraic equations.

In the following section we explain the reason for choosing the domain of high-school algebra. Next, we present a sample of text generated by the implemented system. In section 1.3 we review the available text generation techniques with emphasis on their ability to produce text containing MTUs, and in section 1.4 we consider Tutoring Systems as a special case of text-generation systems. We conclude this chapter by presenting an overview of this dissertation.

## 1.1 Why Algebra?

The first reason for choosing this knowledge domain is its importance. Skill in manipulating algebraic equations is essential to studies in the sciences and in technical subjects. This material is usually taught in a “follow me” fashion (Davis 1974), where a teacher works out a few examples, and then the students are expected to solve similar questions. This approach to tutoring algebra does little to develop a student’s problem-solving skills.

Although a very talented student may be able to learn algebra by following the solution of equations, a mediocre student’s performance improves when the mathematical symbols are accompanied by explanations in ordinary English (Watkins 1977). However, if these explanations are presented in a tedious and repetitive style, a student may become increasingly annoyed with the teacher and bored. Thus, an effective Intelligent Tutoring System should present the material and exercises in a pedagogically sound manner and express itself with an appealing style.

Finally, a rather pragmatic reason for choosing the domain of high school algebra is the easy availability of experts in this area among the UCLA faculty and students.

## 1.2 Input and Output of FIGMENT

We present a stylized version of a sample input to FIGMENT's text-generation components in Fig. 1.3, to illustrate the kind of behaviour it attempts to emulate (the actual input appears in Appendix 8.3). From this input the implemented system produces the following text:

Let us look at a rather interesting topic, namely third degree equations, which is also challenging. Unfortunately, we shall not examine a general technique for solving equations in this subject. However, we can solve certain types of third degree equations by factoring out common factors, or, alternatively, applying the appropriate factorization formula. Here is an equation:

$$x^3 - x^2 - x + 1 = 0$$

First, since  $x^2$  is a factor common to the first and second terms, we factor it out. As you know, we perform this operation hoping to get a factor common to the rest of the terms. Through it we get the following result:

$$x^2(x-1) - x + 1 = 0$$

Next, we rewrite  $-x+1$  as  $-(x-1)$ , arriving at the result we were hoping for:

$$x^2(x-1) - (x-1) = 0$$

Afterwards, we factor out  $x-1$ , yielding:

$$(x-1)(x^2 - 1) = 0$$

We continue by applying the factorization formula  $a^2 - b^2 = (a+b)(a-b)$  to  $x^2-1$ , arriving at the following result:

$$(x-1)^2(x+1) = 0$$

We obtain the solution by solving separately for each factor.

TOPIC:	third-degree
METHOD:	general specific
EQUATION:	$x^3 - x^2 - x + 1 = 0$
(ALTERNATIVE 1)	
RULE:	factor out $x^2$ from terms 1 and 2
PATTERN:	$x^2$ is a factor common to terms 1 and 2
EXPECTATION:	result has common factor with rest of terms
RESULT:	$x^2(x-1) - x + 1 = 0$
RULE:	rewrite $-x+1$ as $-(x-1)$
RESULT:	$x^2(x-1) - (x-1) = 0$
RULE:	factor out $x-1$
RESULT:	$(x-1)(x^2 - 1) = 0$
RULE:	apply factorization formula $a^2 - b^2 = (a+b)(a-b)$
RESULT:	$(x-1)^2(x+1) = 0$
CONTINUE:	product of factors

Fig. 1.3: Stylized Version of Sample Input to FIGMENT's Text-Generation Components

### 1.3 Text Generation Techniques

The issues involved in the generation of MTUs have not been addressed, although a continuum of text generation techniques is already in general use. At one end of the spectrum, a *previously prepared* or *canned* text is displayed. At the other



end, we find text *produced* by a direct translation of knowledge structures (Mann 1981). The middle ground is covered by text generation techniques which store whole phrases (possibly parametrized), and combine them by means of rhetorical rules (Kukich 1983).

The simplest and most commonly used way to have a computer system produce text is for the implementors of the system to figure out in advance what sort of English output will be required and then store it as text strings. The computer merely displays portions of text that has been stored to match a given conversational situation. Clearly, with this technique, the generated text can be very elegantly written. However, the technique has several drawbacks.

i. Lack of consistency.

Revision of a knowledge structure supporting the program is not automatically accompanied by appropriate changes in the text. Therefore, there is no guarantee that the program does what it says it does.

ii. All questions and answers must be anticipated.

iii. Lack of closure.

The computer does not have a conceptual model of what it is saying. As far as the computer is concerned, one text string looks like any other. Therefore the task of generating text does not become easier as more text is generated.

iv. Insufficient individualization.

Because the text is prepared in advance, it is not tailored to the particular needs of each student. This situation can be partially remedied by inserting slots into the canned text that can be filled out by different words according to

the circumstances. This strategy, however, can only account for a small fraction of the cases that arise in a dialogue.

v. Poor connectivity between sentences.

Strings of text have to be physically juxtaposed in cases where it is necessary to generate more than one instance of canned text. This often results in a rather awkward discourse, since the machine lacks a representation for the relationship between the various sentences in the text.

The opposite approach for providing English output produces text by translating knowledge structures of the program directly into English (Davey 1979, Mann and Moore 1980, McDonald 1980, McKeown 1982, Swartout 1982, etc).

Mann et al. (Mann 1981) assert that a competent text generation facility must include the following four components:

- i. A comprehensive, linguistically justified grammar.
- ii. A knowledge representation formalism that can encode diverse kinds of information — This formalism has to suitably represent knowledge like: time, space, events and actions, cause, uncertainty, obligation and modality, etc, in addition to the subject of the discourse.
- iii. A model of the reader — In order to generate acceptable text, the generator must take into account the reader's knowledge. Therefore the model of the reader has to contain information that includes what is obvious and what has already been told.
- iv. A model of the discourse structure and control — We need to model the

interaction between the sentences in text, in order to generate multisentential text. At a higher level, sequences of sentences and paragraphs must be organized in a principled way.

Since the structures being translated are the same ones used in the program's reasoning process, consistency can be ensured. Closure can be realized, because transformations are written to handle large classes of knowledge structures. The problem of individualization can be tackled by taking into consideration the variables which make each piece of dialogue unique, such as the model of the reader or the record of the ongoing dialogue. Finally, discourse-generating strategies can be applied to the knowledge structures in order to attain the desired connectivity of sentences.

The following example (from McKeown 1982) illustrates the process of generating English sentences by translating the knowledge structures of the program. A U.S. Navy toy database, which contains information about vehicles and destructive devices, generates the following answer to the query "DEFINE SHIP":

The ship is a water-going vehicle that travels on the surface. Its surface going capabilities are provided by the DB attributes DRAFT and DISPLACEMENT.

The first step in answering this query is to partition off a subset of the knowledge base that contains information relevant to the question. Next, the resulting subset is matched against predicates that represent the types of information that could appear in a definition — identification, constituents, evidence of a fact or analogies. In this example, the matched predicates are the identification predicate (which accounts for the first sentence), and the evidence predicate (which accounts for the

second one). In the final stage, a tactical component determines the surface ordering of the constituents of these propositions and the grammatical constructs to be used. This component then translates the resulting propositions into English by means of a grammar.

Even though this approach solves the problems present in the canned text technique, it introduces some problems of its own. Since the transformations performed on the knowledge structures are relatively simple, the quality of the text depends to a great extent on the structure of the knowledge. This requires the knowledge itself to be structured, so that it is readily understood, which is not always easy to accomplish in certain domains. Moreover, systems employing this technique have to command considerable linguistic knowledge in order to produce high quality text. Thus far, the available systems have produced text that, although readable, is not graceful. This is due partly to the scarcity of MTUs. In fact, the above mentioned systems have produced few MTUs like "however," "or," "next" or "therefore," which directly reproduce the speaker's organization of the subject matter. However, they do not generate MTUs which reflect the state of the dialogue or the relationship between the speaker and the listener.

#### **1.4 Tutoring Systems**

Tutoring Systems, also known as Computer Assisted Instruction (CAI), can be viewed as a class of text generation facilities in which the communication is two-way, as in a dialogue between a teacher and a student.

Two kinds of tutoring systems are in general use. In the first kind, which is commercially available, the computer functions as a page turner (Basic Algebra, from

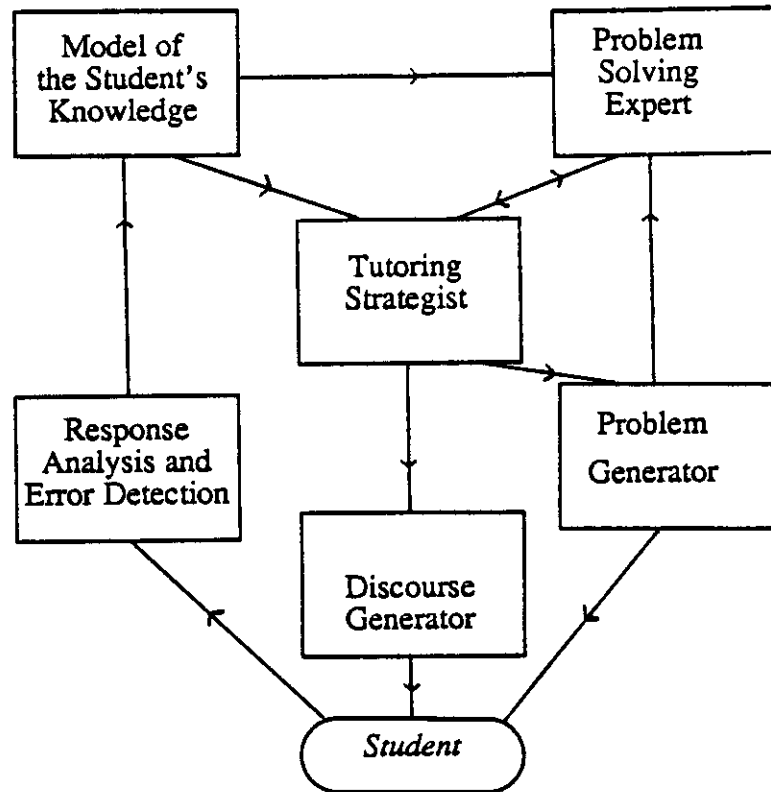
Dorsett, Educational Systems Inc., 1981): Large chunks of canned text are presented interspersed with questions to which the user may answer either YES or NO. The material does not vary on the basis of a student's performance.

In the second kind of tutoring systems, the material presented depends on the user's responses. As in any text generation situation, this material can be either canned text or text generated by direct translation of knowledge structures. In the canned text type of system, a group of teachers attempts to anticipate every wrong response, specifying branches to appropriate remedial material, based on their ideas about the student's misconceptions (Bork 1981). Systems generating text by direct translation of knowledge structures have, as the canned text system, a representation of the subject matter they teach. However, unlike the former system, direct translation systems are able actually to diagnose student's misconceptions from his mistakes and respond to his individual needs. This type of system is known in the literature as Intelligent CAI (ICAI) or Intelligent Tutoring Systems (ITS).

#### **1.4.1 Intelligent Tutoring Systems**

Intelligent Tutoring Systems have been developed in several domains of knowledge in the past few years, including medical diagnosis, problem-solving skills and electronic trouble shooting. These systems feature diverse methods of instruction, taking the form of coaches, like WEST (Burton and Brown 1981) and WUMPUS (Goldstein 1981); consultants, such as A Consultancy System for MACSYMA (Genesereth 1978, 1981); problem solving monitors, like GUIDON (Clancey 1979, 1981a-c); and laboratory instructors, like SOPHIE (Brown, Burton and deKleer 1981).

Most Intelligent Tutoring Systems contain all the modules present in a general text generation facility, plus additional components specific to the tutoring environment. The first four modules we will consider, are common to text generation and tutoring systems, while the last two modules are particular to tutoring systems only (see Fig. 1.4).



**Fig. 1.4: Block diagram of an Intelligent Tutoring System**

i. **Problem Solving Expert.**

This module contains the knowledge that the system tries to impart to the student, which is in our case a few chapters of high-school algebra. The knowledge representation formalisms of an ITS have to be able to encode the distinctive features of the subject, e.g., to describe an equation, the purpose of

a step in the solution, or the correctness of an operation.

ii. **Model of the Student's Knowledge.**

This module contains the information required by a general text generation system, namely what is obvious, what has already been told, etc. The two methods generally used in order to model the knowledge of the student are the overlay method in which the student model is a subset of the expert's knowledge base (Goldstein 1981), and the differential model that abstracts how the student's behaviour is critically different from that of the expert (Burton and Brown 1981).

iii. **Tutoring Strategist.**

This module specifies how the system presents its material to the student. It contains pedagogical rules for deciding which material to present and in what order. Thus, it performs the high-level discourse-organization function mentioned in the Models-of-the-Discourse component of a text generation facility.

iv. **Discourse Generator.**

This module is used for generating the explanations, questions and remarks presented by the system during the dialogue with the student. In order to generate text that is not awkward or misleading, this module should model the interaction between sentences, performing the low-level function of the Model-of-the-Discourse component. One of the results of this model is the generation of appropriate MTUs. In addition, this module should be able to control a variety of language effects at the sentence level. These effects are produced by the arrangements of the words used.

v. Problem Generator.

This module presents problems (exercises and examples) to the student. It usually accepts specifications regarding the level of difficulty of the problem, the elements of the material to be included in the problem, etc (Sleeman 1981b).

vi. Response Analysis and Error Detection.

In this module the user's response is analyzed in order to determine whether it is correct or not. If incorrect, this module should be able to trace the user's misconception responsible for the mistake. In the subject of algebraic equations, error detection has been performed by means of *mal-rules*. Given a set of rules used to solve an equation, a mal-rule is a perturbation of a good rule, which students use instead of the good rule (Brown and VanLehn 1980, Sleeman 1981a, 1982).

Each of the above mentioned systems emphasizes some of the aspects of the overall teaching system and neglects others. However, they all concentrate on conveying the curriculum material, mostly disregarding the *manner* in which this material is delivered. All the tutoring systems have some provision for encouraging the student, even the ones that present canned text. WUMPUS, WEST and GUIDON also feature tutorial rules for guiding the discussion and choosing explanations. But even in these systems, the tutorial rules control *what to say*, not *how to say it*. No attempt is made to tailor the generated text (as opposed to the material presented) to the student's individual needs, i.e., no variation is introduced into the text, the parts that deserve special attention are not emphasized, a repeated mistake is not dealt with differently on each occasion, etc. Thus, even though the generated text is readable, it is rather tedious and stilted.



## 1.5 Overview of this Dissertation

In this research we discuss the knowledge required by a text generation system and the processes it activates in order to generate discourse containing MTUs. Let us now briefly describe the contents of this dissertation.

Chapter 2 presents a taxonomy of meta-technical utterances according to their function as seen by a tutor in transmitting the subject matter to the student. This classification distinguishes between three types of MTUs: (1) Knowledge Organization, (2) Knowledge Acquisition, and (3) Affect Maintenance. Knowledge Organization MTUs reflect the organization of the material in the tutor's mind, e.g., "next," "however"; Knowledge-Acquisition MTUs provide information that enables a student to prepare adequate mental resources, e.g., "as I said before," "let us consider"; and Affect-Maintenance MTUs inform a student of favourable and unfavourable events, e.g., "unfortunately," and contribute to maintaining his positive attitude, e.g., "I shall go over this explanation again."

Chapter 3 discusses the overall design of FIGMENT. This design enables an Intelligent Tutoring System to produce discourse which contains MTUs. We also examine the strategic components of FIGMENT. These are the modules which determine the material to be presented, namely the Problem Solving Expert, the Model of the Student's Knowledge and the Tutoring Strategist.

Chapters 4, 5 and 6 examine in detail the tactical component of FIGMENT, namely the Discourse Generator. This module is divided into two sub-components, the Comprehension-Processes Module, which determines the need for MTUs, and the Sentence Composer, which organizes the output of the Comprehension-Processes

Module into paragraphs and sentences, and translates it into English.

Chapter 4 describes the generation of Knowledge-Acquisition MTUs and of the Affect-Maintenance MTUs which are issued to maintain a positive attitude in a student. The Comprehension-Processes Module generates codes which reflect requirements for these types of MTUs, by consulting simplified models of the comprehension processes of a student. These models simulate some mental activities performed by a reader or listener upon encountering technical messages. In section 4.6 we present a partial output of the Comprehension-Processes Module, and describe how it was obtained.

Chapter 5 contains a detailed discussion of the generation of Knowledge-Organization MTUs and of the Affect-Maintenance MTUs which inform a student of the nature of the events under consideration. These MTUs are directly derived from the organization of the knowledge of a tutor, and from the sequence of events which took place while solving an equation. Section 5.6 contains a complete output of the Comprehension-Processes Module, accompanied by a description of the manner in which it was generated. It also features the text produced by the Sentence-Composer for this output.

Chapter 6 describes the structure and operation of the Sentence Composer. This module translates the technical messages and accompanying MTU requirement-codes into English by using a phrasal dictionary and applying rhetorical rules.

Finally, chapter 7 discusses the importance of the ideas of this dissertation and potential avenues for future research.

## CHAPTER 2

### Semantics of Meta-Technical Utterances

In this chapter we first present typical expressions used in math tutoring, and focus on their MTUs. We then discuss linguistic research performed on the semantics of meta-lingual utterances. Finally, we present a functional taxonomy of MTUs, on which the design of FIGMENT is based.

#### 2.1 Sample Expressions Used in Tutoring Algebra

The MTUs in the following expressions transmit information to improve a student's understanding of the algebraic operations and entities in question:

1. **“We shall perform algebraic expansion and then collect terms.”**

The MTU in boldface depicts a *temporal* relation, i.e., that the actions (algebraic expansion and term collection) must be performed sequentially.

2. **“We cross-multiply in order to get rid of fractions.”**

The conjunctive expression “in order to” indicates that the latter portion of the sentence contains the *purpose* for which the action described in the former portion is performed.

3. **“We factored out  $(x-2)$ , hoping to find a common factor with the rest of the equation. Unfortunately, however, we have been unable to find such a factor.”**

These MTUs convey two types of relations. The word “hoping” signals that the actions described at the beginning of the first sentence purport to achieve the result described in the second part of this sentence, and that this result is uncertain. “Unfortunately, however” represents *violation* of these hopes.

4. **“The following example illustrates ... .”**

This sentence declares that the sentence or group of sentences following it will contain an instantiation of the general material presented earlier.

5. **“An alternative way of solving this equation consists of ... .”**

6. **“Incidentally, there is a general method for solving quadratic equations.”**

The MTU “incidentally” alerts the listener that the forthcoming sentence contains a *digression*, i.e., that the focus of the discussion is about to shift temporarily.

7. **“You take the first two terms, namely  $3x$  and  $x^2$ , and ... .”**

The MTU “namely” indicates that the forthcoming list of elements is an explicit *paraphrase* of the previous implicit description.

8. **“Let us now consider the topic of linear equations.”**

The expression in boldface indicates that discussion on a previous subject has been completed, and that a different topic is about to be discussed.

9. **“As I have said before, this type of equation is solved by ... .”**

The MTU in boldface signals to the student that the tutor is aware of having previously presented the forthcoming information.

10. "The solution to this equation is quite complicated."

This sentence informs the student of a particular attribute of the equation in question.

The MTUs featured in expressions 1-8 portray relationships between a forthcoming sentence or group of sentences and preceding discourse, while the MTUs in expressions 9 and 10 are associated only with forthcoming text. While most of the MTUs presented in examples 1-7 have been thoroughly studied by linguists, MTUs like the ones in examples 8-10 have been neglected despite their frequent use.

## 2.2 Overview of Related Research

The relations between clauses or predications have long been of interest to linguists. As early as 1668, Wilkins devised a taxonomy of the English Language which included conjunctions and conjunctive expressions such as "although," "indeed," "therefore" and "for example" (see Appendix 1). His classification of conjunctions is based on their function, e.g., interrogative, declarative, causal, etc.

Recent linguists have examined the relations in the English language from two opposite points of view. Sweet, Winter, Quirk and Hallyday & Hasan have focused on conjunctions, also denoted *surface* conjunctions; whereas Grimes and Longacre have investigated the *deep* structures underlying interclausal relations, e.g., causal, temporal, illustration, introduction, deixis, etc. The English representation of most of these structures includes surface conjunctions and conjunctive expressions. However, some deep structures do not feature conjunctive expressions. For instance (from Longacre 1976):

AWARENESS ATTRIBUTION	—	“John knows that Mary is coming”
SPEECH ATTRIBUTION	—	“John said ‘Mary is coming’”
DEIXIS (INTRODUCTION)	—	“There was a man. The girls gave him food”
DEIXIS (IDENTIFICATION)	—	“He bought an apple and that was what he ate”

In 1891, Sweet investigated the function of conjunctions and half-conjunctions (i.e., independent adverbs such as “nevertheless” or “still”), and classified them according to their role as sentence connectors. Taxonomies performed by contemporary linguists have not strayed far from Sweet’s original classification, a fact that prompts us to examine Sweet’s taxonomy in detail and use it as a basis for further discussion.

i. Affirmative or copulative.

These conjunctions connect without implying any special kind of connection, e.g., “and,” “also,” “too.” More emphatic forms include: “both ... and,” “not only ... but,” “furthermore,” “moreover,” “now” and “well.”

ii. Alternative.

The chief alternative conjunction is “or,” whose emphatic form is “either ... or.”

iii. Negative.

The chief negative conjunction is “neither ... nor.”

iv. Adversative.

These conjunctions add something which is unexpected, does not follow naturally from what has just been said, or seems to halt the natural progress of the narration. Some conjunctions and half-conjunctions used as adversatives are: “but,” “still,” “nevertheless,” “however,” “at the same time” and “in spite of that.”

v. **Concessive.**

These conjunctions imply that the forthcoming sentence is followed by one with adversative meaning, e.g., “though ... yet,” “although ... but.” Whereas adversative conjunctions refer backwards, concessive conjunctions refer forwards.

vi. **Hypothetical.**

The chief hypothetical conjunction is “if.” Other hypothetical conjunctions include “in case,” “suppose,” “supposing that” and “provided that.” “Unless” which equals “if not,” is a hypothetical negation. A hypothetical difference is expressed by “otherwise” (meaning: if we act differently). The correlative pair “whether ... or” expresses an alternative hypothesis, as in: “He will have to do it, whether he likes it or not.” Hypothetical concession is expressed by “even if” (e.g., “Even if he is mistaken, you need not tell him so”); hypothetical comparison is expressed by “as if” (e.g., “He started as if he had been shot”).

vii. **Temporal.**

These conjunctions include “when,” “as,” “while,” “before,” “after,” “since,” “until” and “till.” Some temporal conjunctions which express immediateness are “directly,” “as soon as,” “just as,” “just after.”

viii. **Causal.**

These conjunctions are subdivided into three classes: cause (e.g., “because,” “for,” “since,” “as”), effect (e.g., “therefore,” “so,” “then,” “accordingly,” “consequently”) and purpose (e.g., “in order (that),” “so that ... not”).

Winter (1968), Quirk et al.(1972) and Hallyday & Hasan (1976) also classified interclausal relations according to the semantics of surface conjunctions and conjunctive expressions. The classification arrived at by Hallyday and Hasan is the most refined and complete of the three. They considered types of conjunctive expressions similar to the ones reviewed by Sweet, and devised a classification scheme composed of four main categories. Additive contains Sweet's affirmative and alternative categories; Adversative collapses Sweet's adversative, concessive and hypothetical-difference categories; Causal and Temporal.

In addition, they distinguished between relations inherent in the phenomena under consideration, and those in the communication process between the speaker and listener (*external vs. internal*). The following example illustrates these concepts:

- a. First he switched on the light. Next he inserted the key into the lock.
- b. First he was unable to stand upright. Next he was incapable of inserting the key into the lock.

In (a) two time-related events are described, whereas in (b) there are no actual events, but only *linguistic* events, for the time sequence is in the speaker's organization of his discourse.

Hallyday and Hasan further refined their categories, with internal subclassifications such as: afterthought ("incidentally," "by the way") and exemplificatory ("for instance," "thus") in the additive category; avowal ("in fact," "actually"), correction ("instead," "rather") and dismissal ("in any case," "either way") in the adversative category, etc. They also presented other cohesive items which are not part of any of the conjunctive relations identified in their classification:



“now,” “of course,” “well,” “anyway,” “surely” and “after all.” A summary table of their taxonomy is reproduced in Fig. 2.1.

The taxonomies of interclausal relations performed by Grimes (1975) and Longacre (1976) are based on deep structures, rather than on surface conjunctions. Some of the categories in their classifications do not contain surface conjunctions, and will not be addressed here.

Longacre’s classification is based on the fundamentality of the interclausal relations to the communication process. It contains the following main headings: Basic, Elaborative and Frustration.

- i. Basic relations are essential to the structure of discourse. These are none other than the relations introduced by Sweet, but collapsed into four categories, namely:
  - (1) Conjoining, e.g., “and,” “but,” “except” and “as much as ... ”;
  - (2) Alternation, e.g., “either ... or”;
  - (3) Temporal, e.g., “while,” “as,” “after” and “then”; and
  - (4) Implication, which includes Sweet’s causal and hypothetical categories.
  
- ii. Elaborative relations contain embellishments or rhetorical devices of the language. They are separated into the following categories:
  - (1) Paraphrase, including generic-specific (“to be more specific”), specific-generic, summary, preview and negated antonym (“it is not hot, but warm”);
  - (2) Illustration, including exemplification (“for example”) and simile (“she acts like a baby”);
  - (3) Deixis; and

	External/Internal	Internal (unless otherwise specified)		
Additive	Additive, simple: Additive <i>and, and also</i> Negative <i>nor, and ... not</i> Alternative <i>or, or else</i>	Complex, emphatic: Additive <i>furthermore, in addition, besides</i> Alternative <i>alternatively</i>  Complex, de-emphatic: After-thought <i>incidentally, by the way</i>	Apposition: Expository <i>that is, I mean, in other words</i>  Exemplificatory <i>for instance, thus</i>	Comparison: Similar <i>likewise, similarly, in the same way</i>  Dissimilar <i>on the other</i>
Adversative	Adversative 'proper': Simple <i>yet, though only</i> Containing 'and' Emphatic <i>however, nevertheless, despite this</i>	Contrastive: Avowal <i>in fact, actually, as a matter of fact</i>  Contrastive (external): Simple <i>but, and</i> Emphatic: <i>however, on the other hand, at the same time</i>	Correction: Of meaning <i>instead rather, on the contrary at least, rather, I mean</i>  Of wording	Dismissal: Closed <i>in any case, in either case, whichever way it is</i>  Open-ended <i>in any case anyhow, at any rate however it is</i>
Causal	Causal, general: Simple <i>so, then, hence, therefore</i> Emphatic <i>consequently, because of this</i>  Causal, specific: Reason <i>for this reason, on account of this</i> Result <i>as a result, in consequence</i> Purpose <i>for this purpose, with this in mind</i>	Reversed causal: Simple <i>for, because</i>  Causal, specific: Reason <i>it follows, on this basis</i> Result <i>arising out of this</i> Purpose <i>to this end</i>	Conditional (also external): Simple <i>then</i> Emphatic <i>in that case, in such an event, that being so</i>  Generalized <i>under the circumstances otherwise, under other circumstances</i>  Reversed polarity	Respective: Direct <i>in this respect, in this regard, with reference to this</i>  Reversed polarity <i>otherwise, in other respects, aside from this</i>
Temporal	Temporal, simple (external only): Sequential <i>then, next, after that</i> Simultaneous <i>just then, at the same time</i> Preceding <i>previously, before that</i>  Conclusive: Simple <i>finally, at last</i>  Correlative forms: Sequential <i>first ... then</i> Conclusive <i>at first ... in the end</i>	Complex (external only): Immediate <i>at once, thereupon</i> Interrupted <i>soon, after a time</i> Repetitive <i>next time, on another occasion</i> Specific <i>next day, an hour later</i> Durative <i>meanwhile</i> Terminal <i>until then</i> Punctiliar <i>at this moment</i>	Internal temporal: Sequential <i>then, next, secondly</i> Conclusive <i>finally, in conclusion</i>  Correlative forms: Sequential <i>first ... next</i> Conclusive <i>... finally</i>	'Here and now': Past <i>up to now, hitherto</i> Present <i>at this point, here</i> Future <i>from now on, hence-forward</i>  Summary: Summarizing <i>to sum up, in short, briefly</i> Resumptive <i>to resume, to return to the point</i>

Fig. 2.1: Summary Table of Conjunctive Relations by Hallyday and Hasan

(4) Attribution.

iii. Frustration relations express the violation of expectations established by some of the subclassifications in the Basic and Elaborative headings. The following examples illustrate this category:

She is smart but pretty.	(frustrated coupling)
They left but did not arrive.	(frustrated succession)
He drives but does not look.	(frustrated overlap)
Even if ... , I will (not) ...	(frustrated hypothesis)
He was poisoned, but did not die.	(frustrated cause)
I thought you knew, but you did not.	(frustrated attribution)

Grimes' first division is performed along the dimension of coordination and subordination of rhetorical predicates. Nevertheless, most of his final categories resemble those of Longacre.

Winter (1977) expands on these works by considering the notion of *predictability of discourse structure* in relation to the semantics of sentence connection. In reference to the conjunctions and conjunctive expressions presented above, he states: "One of the most important connective functions of this vocabulary is that the presence of one of its items in a particular sentence can *signpost* what kind of information is to be presented in the sentence or sentences which immediately follow it. Such a signposting function will be called *anticipation*."

Farnes (1973) claims that the identification and use of *link or function words*, such as "although," "rather," "whereas" and "similarly" greatly aids comprehension. In addition, he broadens the concept of signaling to include entire clauses and sentences whose main function is the clarification of the discourse structure (e.g., "There are three points to consider"). Finally, he notes that this signaling is usually performed cataphorically, i.e., in advance.

Reichman (1978, 1984) focuses her research on the generation of coherent dialogue. Her views are similar to Farnes' regarding the importance of *clue words* and some full phrases in signaling *context transitions*. She expands the concept of signaling to shifts in verb tense, changes in mode of reference (e.g., a shift from pronominalization to explicit reference signals a change of context), etc.

Finally, Hoey (1979) considers signals similar to the ones researched by Reichman. However, he claims that they not only signal shifts in context, but also indicate other functions of the discourse structure. For example, the first sentence of any discourse is expected to provide a context for subsequent sentences; a problem is signaled by verbs like "require" or "need to avoid." Furthermore, he points out that problems of comprehension have been shown to arise from faulty or missing signaling.

### **2.3 A Functional Taxonomy for the Generation of MTUs**

The classification of Meta-Technical Utterances presented in this section is based on their function as seen by the tutor in transmitting the subject matter to the student. In our taxonomy we recognize three main functions of MTUs: (1) Knowledge Organization, (2) Knowledge Acquisition and (3) Affect Maintenance. The system whose design is outlined in the next chapter makes use of this taxonomy in order to generate MTUs necessary for the formulation of fluent discourse.

#### **2.3.1 Knowledge-Organization MTUs**

At each point in time, the information residing in a student's (or tutor's) mind can be visualized as a network whose nodes contain the individual information items, and whose links contain the relations between the nodes (Anderson 1980, 1983). For

example, NODE1 contains the *purpose* of NODE2, NODE3 is the *cause* for NODE1, or NODE4 contains the algebraic operation performed *after* NODE2. These relations directly reflect a teacher's knowledge about algebraic equations and about the events which take place during the solution of an equation. The MTUs which express these relations may also convey information about the state of the dialogue. Hence, they roughly correspond to Hallyday and Hasan's external/internal category (in Hallyday and Hasan 1976) and Longacre's basic heading (in Longacre 1976). The Knowledge Organization function consists of transmitting these relations, and is performed by the following types of MTUs:

i. Additive.

These MTUs signal *additional elements in a list* ("and," "also," "nor"); *availability of alternatives* ("alternatively," "or," "or else"); or *realization of expectations* ("indeed").

ii. Adversative.

These MTUs signal *violation of expectations* ("however," "nevertheless," "although," "but," "despite this"); *dismissal*, which is triggered when two different paths in a solution arrive at the same pattern ("in any case," "either way," "at any rate") or *recognition*, which is triggered when a well known pattern, implicit in the equation, is recognized and made explicit ("notice that").

iii. Causal.

These MTUs pertain to the knowledge about the subject matter, and signal the *reason for performing an operation* ("for this reason," "on account of this," "it follows," "therefore," "so," "because of this"); its *purpose* ("for this

purpose," "with this in mind," "to this end"); *expectations* ("hopefully," "expecting to get"); *result* ("as a result," "in consequence"); *means* ("this can be accomplished by"), or *correctness* ("this works because").

iv. **Attributive.**

These MTUs signal either a *generality* or *particularity* relationship of a group of algebraic rules with the subject under consideration. A technique may constitute a *general* way of solving equations of a given topic ("in general," "generally"), or may be able to solve only *certain types of equations* ("particular instances of," "certain types of"). Even though this relationship appears in the internal category of Hallyday and Hasan, we have added it to the Knowledge-Organization MTUs because in the realm of algebra the generality or particularity of a technique does not depend on the status of the conversation, but on its applicability to a topic or type of equation.

v. **Temporal.**

These MTUs signal *sequence* ("then," "next," "after that," "first ... next," "finally," "previously"), or *partial sequence* ("at the same time"). They refer to the events which occur during the solution of an equation.

The reader will note that this taxonomy contains some changes with respect to Hallyday and Hasan's classification (see Fig. 2.1). Due to the conditions imposed by the domain of algebra we have added a relation of realization-of-expectations to the additive category and the recognition relation to the adversative heading. We have also added the attributive classification, and have classified the dismissal relation as external, since it is directly derived from the solution of an equation.

The reader will also note that there exist situations in which the distinction between temporal MTUs and certain additive MTUs may become fuzzy. For instance, consider the following sentences:

1. “We remove parentheses. Next we collect terms,” and
2. “We remove parentheses and collect terms.”

The first sentence contains a temporal MTU, while the second one features an additive MTU. Nevertheless, both sentences convey the same meaning, namely a temporal relationship.

Whereas the values of the above mentioned relations would generally depend on the solution to each particular equation, most of the relations in the causal category are inherent in the solution methods and do not vary from equation to equation. For example, each method has a purpose, a pattern or reason for applying it, a correctness proof, etc. Thus, the knowledge required for commenting on the causal aspects of the solution of equations should preferably reside in a module that contains the expertise about strategies for solving equations. We call this module a Problem-Solving Expert (see section 3.1).

### **2.3.2 Knowledge-Acquisition MTUs**

MTUs that facilitate Knowledge-Acquisition are related to the interaction between the teacher and the student and to the state of the discourse, rather than to the subject matter itself. They ease the assimilation of the subject matter by alerting the student to prepare adequate mental resources. In the context of tutoring algebra, the Knowledge-Acquisition function is performed by the following types of MTUs:

i. Motivational.

These MTUs are often used by a teacher to motivate a student to listen to the forthcoming technical utterance. For example, if a new method is to be taught, the tutor might say: "This method is very expeditious." If a student has to practice the same type of equation many times, the teacher might say: "Third degree equations are rather difficult and demand lots of practice."

ii. Focal.

A student generally attempts to process a forthcoming technical utterance in the currently active focus space (Grosz 1977, 1979). To provoke a change in the active focus space or to close an open focus space, a teacher must present the student with an MTU to this effect. For example, the focal MTU "Let us now consider the following equation" will close the focus space corresponding to the previous equation and open a new focus space for the next equation. In addition, the MTUs corresponding to Sweet's hypothetical category signal *alternatives that should have been attempted* ("If you had ... you would have ... ") or *mistakes that could have been avoided* ("Instead you ... "). Temporary focus shifts are signaled by MTUs like "incidentally" or "by the way." In terms of the above-mentioned network analogy, focal MTUs create a space to contain a forthcoming sub-net.

iii. Categorical.

To clarify the relationship between a forthcoming technical utterance and a specific item in active focus, a tutor might say: "Let's take the first term on the right hand side, namely  $x^2(x - 3)$ , ... ." The MTU "namely" alerts the listener to the fact that the symbolic description paraphrases the preceding



positional description and is not to be added to the first term on the right hand side. Categorical MTUs are included in the taxonomies presented in Hallyday and Hasan's internal classification (in Hallyday and Hasan 1976) and in Longacre's elaborative heading (in Longacre 1976). Some types of MTUs which belong to this subclass are: *paraphrase* ("in other words"); *specific-generic* ("more generally"); *generic-specific* ("to be more specific"); *exemplification* ("for example"); and *avowal*, which signals that the forthcoming technical utterance contradicts what the current state of the discourse would lead us to expect ("as a matter of fact," "in fact"). These MTUs specify the manner in which a student should use the information in the forthcoming technical utterance to update the information featured in the previous one.

iv. **Implementational.**

These MTUs guide the selection of a computational activity required for assimilating the technical utterance that follows. We identify two main activities: *adding* an item to one's knowledge pool and *verifying* the workings of existing knowledge for possible revision. For example, if the tutor wishes a student to use the forthcoming statement to verify existing knowledge, he should signal his intent with an MTU like: "As I have stated before, since  $x-3$  is a factor common to ... "; or if the student should update a pattern for an equation, the tutor might say: "This equation is similar to '... ." On the other hand, to prepare a student for learning a new subject (i.e., transfer to addition mode) a teacher might say: "We shall now consider a new type of equation."

v. Estimational.

These MTUs inform the student that the forthcoming technical utterance is of unusual length and/or complexity. They indicate the processing time and computational power the student should prepare to process the forthcoming technical utterance. Examples are: “This equation is rather straightforward” or “The following method entails several computations.”

In order to illustrate the importance of these Knowledge-Acquisition MTUs, let us examine the following imperfect discourse:

*“The method of completion to square for solving quadratic equations works as follows”*

... description of completion to square method ...

... examples of application of this method ...

*“In order to perform completion to square we have to execute the following steps”*

... description of completion to square method ...

The dissonance in the preceding discourse stems from the repetition of preparatory directives in the first and fourth lines. The first time the completion to square method is introduced, the MTU “*The method of completion to square ... works as follows*” performs the dual role of establishing our expectations for receiving new knowledge and of signaling that the forthcoming technical utterance is a description of a method already in active focus. The MTU in the fourth line repeats the directives given first, even though their English representations differ. We again prepare ourselves to add the method description to our knowledge pool. But upon discovering that this information is already in our knowledge pool, we experience the discomforting feeling that the discourse generator has no recollection of previous activities.

Notice, however, that had the fourth line been replaced by "*Let's go over the method of completion to square again,*" the learning process would be much smoother; we would prepare to use the incoming information for verifying already existing knowledge, instead of treating it as new information to be added on.

This example supports the claims made by Hoey, Farnes and others (see section 2.2) regarding the adverse effects of the improper usage of MTUs on the learning process. We need a module that represents the influence of both technical and meta-technical utterances on the listener's mental activities, in order to generate a commentary which includes Knowledge-Acquisition MTUs. This module would then inspect the technical utterances about to be issued, determine their effect on the comprehension processes of the listener, and generate adequate Knowledge-Acquisition MTUs. We shall call this module Comprehension-Processes Module (see Chapter 4).

### 2.3.3 Affect-Maintenance MTUs

One of a tutor's goals is to teach a student which algebraic operations and results are considered favourable and which are considered unfavourable. In addition, a tutor wishes the attitude of a student to remain positive throughout the session. To achieve these goals, a tutor may be required to use Affect-Maintenance MTUs. Since these MTUs have no direct role in signaling inter-clausal relations, they have been neglected in the taxonomies reviewed in section 2.2. Nevertheless, it is hard to dispute their importance in a tutoring environment. We divide these MTUs into two subclasses according to their goals.

- i. Affect-Transference.

These MTUs prepare a student for a forthcoming technical utterance which

might have an affective impact. Examples of these MTUs are “unfortunately” and “fortunately.” For example, if the teacher says: “Unfortunately, the only way of solving this equation is to remove parentheses and collect terms,” the student should understand that this method of solution is considered undesirable and should be used only as a last resort. This point might be missed by the student had the MTU “unfortunately” been omitted.

Affect-transference MTUs are sometimes used by a tutor to alert a student in advance of an upcoming utterance, even when its affective impact is quite obvious. For instance, if the result of an algebraic operation violates a previous expectation, thereby causing failure to solve the equation under consideration, a tutor might say: “Unfortunately, contrary to our expectations, we arrive at the following result.” If a particular method fails to solve an equation and there exists another alternative, the following sentence could be generated by the tutor: “Fortunately, however, there is another way of solving this equation.”

ii. Consolatory.

These MTUs are applied in situations where the goal of maintaining a positive student attitude throughout a tutorial session cannot be accomplished by using Knowledge-Acquisition MTUs. For example, a student may fail to understand a solution method in spite of having received preparatory Knowledge-Acquisition MTUs. In such cases, a teacher should reassure and console the student, so that he will be able to continue learning. This is the purpose of consolatory MTUs such as: “I know these are many alternatives, but you will benefit from learning the techniques they illustrate,” or “Don’t worry, I will

explain this a few more times.’’

The reader will notice that some of the Affect-Maintenance MTUs are equivalent to or contain Knowledge-Acquisition MTUs (in particular estimational MTUs). The reason for this is that an MTU may perform several illocutionary acts (Appelt 1982), e.g., it can signal to the student which mental resources to prepare and, simultaneously, dispel negative affects.

To generate commentaries which have a desired affective influence on a student, a discourse generator needs to recognize the affective impact of both technical and meta-technical utterances. It has to neutralize anticipated negative affects by generating adequate Affect-Maintenance MTUs. Since consolatory MTUs are issued to alleviate negative affects caused by a failure to assimilate the subject matter, their generation is intimately connected to the generation of Knowledge-Acquisition MTUs. The generation of affect-transference MTUs, on the other hand, is linked to transmission of information and to the realization or violation of a student's previous expectations.

#### **2.3.4 Summary of Meta-Technical Utterances**

Fig. 2.2 contains a summary of the taxonomy presented above.

Knowledge Organization	{ Additive Adversative Causal Attributive Temporal	<i>and; or; indeed however; either way; notice that therefore; in order to; as a result in general; certain types of then; next; second</i>
Knowledge Acquisition	{ Motivational Focal Categorical Implementational Estimational	<i>this topic is very important let us consider the following for example; namely as I said before this equation is quite difficult</i>
Affect Maintenance	{ Affect-Transference Consolatory	<i>fortunately; unfortunately don't worry, I will explain it again</i>

**Fig. 2.2: Summary of Meta-Technical Utterances**

## CHAPTER 3

### Design of FIGMENT

This chapter outlines a system designed to generate fluent and cogent commentaries on algebraic equations, based on the taxonomy presented in Chapter 2. The generation of a commentary is performed in three stages (see Fig. 3.1).

- i. In the first stage, the strategic components of FIGMENT produce a *technical file*, which consists of a list of *technical messages*. These components are (1) Problem Solving Expert, (2) Model of the Student's Knowledge and (3) Tutoring Strategist. The Problem Solving Expert solves the equation, and produces a tree-like graph in which each branch contains an attempted solution alternative. Next, the Tutoring Strategist modifies this graph by suppressing alternatives and steps which are well-known to the student and by adding explanations where necessary, such as for the purpose of an operation or for its description. Both modules use information about the state of the student's knowledge provided by the Model of the Student's Knowledge.
- ii. In the next stage, the Comprehension-Processes Module complements and revises the technical file by adding appropriate Meta-Technical Utterances. The affect-transference MTUs and most Knowledge-Organization MTUs can be directly derived from the structure of the technical file. The Knowledge-Acquisition MTUs and the consolatory MTUs are generated by simulating some of the comprehension-processes activated by the student when reading

or listening to an explanation.

- iii. In the final stage, the Sentence Composer organizes the completed message into paragraphs and sentences and translates it into English. Some Knowledge-Organization MTUs which depend on the final structure of the text are generated at this stage.

This text generation process is based on a clear division between the strategic and tactical components, where the results of the strategic components are completely determined before being passed to the tactical component. Nevertheless, the basic tenets of this research could also accommodate a control structure like the one suggested by Appelt (1982), which allows backtracking between the tactical and the strategic components.

In the following sections we consider the modules that are used for the first stage.

### **3.1 Problem Solving Expertise Module**

The Problem Solving Expertise Module (PSE) is an expert system for solving algebraic equations. It is designed along accepted guidelines (Hayes-Roth 1983, Waterman 1986), and is composed of two main parts: a domain knowledge and a problem-solving knowledge. At present, only the domain knowledge component has been implemented. In particular, attention has been focused on the information required to perform the text generation task.



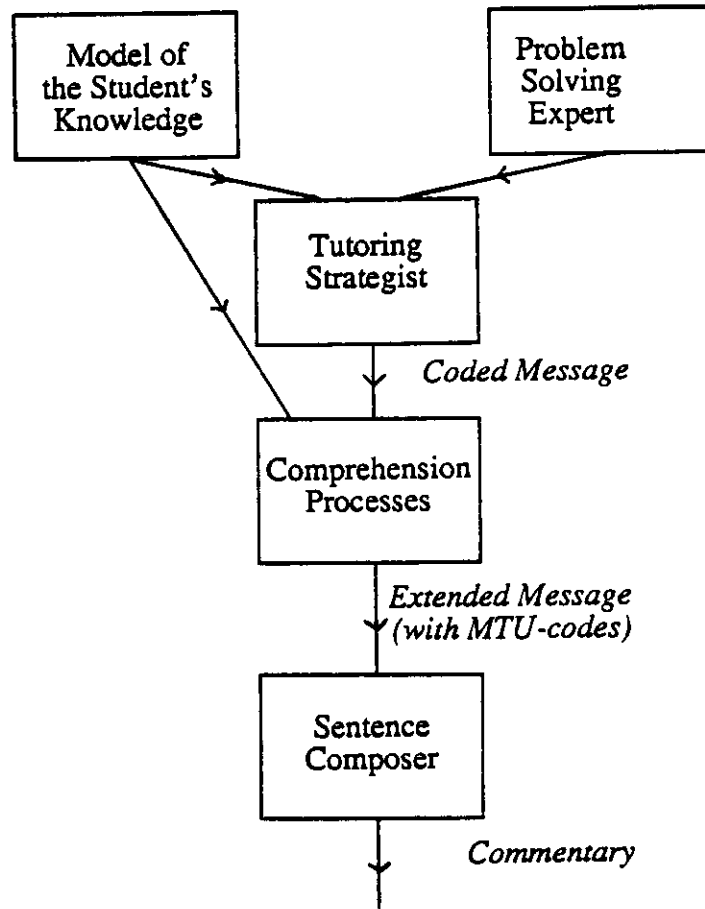


Fig. 3.1: Block Diagram of FIGMENT

### 3.1.1 Domain Knowledge

The Domain Knowledge of the Problem Solving Expertise module has a hierarchical representation (Anderson 1980, 1983). The top level of the hierarchy contains the different topics in the domain of high-school algebra, e.g., linear and quadratic; the next level contains equations which have frequently encountered patterns, e.g., " $ax + b = c$ " and " $ax^2 + bx + c = 0$ "; and the third level contains the different methods which can be used to solve equations, e.g., factorization, substitution, removal of parentheses and collection of terms.

In order to produce a cogent commentary on the solution of an equation, FIGMENT requires an augmented representation of the domain knowledge (Clancey 1979, 1981a-c). In addition to the knowledge used to solve equations, this representation contains knowledge required to perform the text generation task. The selected representation reminds us of the Frame formalism, in that slots are used to store the relevant information (Minsky 1975, Fikes and Kehler 1985). For the first two levels of the hierarchy, the domain knowledge appears in the following slots:

i. **Attribute-list.**

This slot contains a list of attributes which are applicable to the topic or equation type under consideration. For example, a topic may be interesting, important or challenging. Each attribute is accompanied by a number which represents its degree (“very,” “rather”). This information may be used by the Comprehension-Processes Module to generate a motivational sentence like the following: “Let us consider the interesting topic of quadratic equations, which illustrates some useful techniques” (see section 4.5).

ii. **General method.**

This slot contains the names of one or more general techniques used to solve equations of this type.

iii. **Specific methods.**

This slot contains the names of one or more techniques which may be applicable under certain circumstances.

iv. **Complexity.**

This slot contains a compound assessment of the complexity of the equations

and techniques typical of the topic in question. The complexity of an equation is a measure of the difficulty of recognizing the actions to be taken in order to solve it. A linear equation of the form " $ax+b=c$ " is considered to be quite simple, since its solution can be achieved with certainty by applying a few straightforward operations, while a quadratic equation of the form " $ax^2+bx+c=0$ " is more complex, since a student would have more difficulty in recollecting the quadratic formula or the method of completion of the square.

Each method at the third level of the hierarchy is represented by an annotated rule, which has two main components, namely *Mathematical and Planning Knowledge* and *Rule Evaluators* (see Fig. 3.2).

The *Mathematical and Planning Knowledge* component contains the mathematical knowledge required to solve an equation and explain its solution. As above, the mathematical knowledge is represented in slots which express the static relationships inherent in the subject matter (such as reason, purpose, expectations, result, means, continuation and possible alternatives).

- i. The slot for Pattern and/or Preconditions for a rule application contains the pattern which is matched to the given equation in order to enable the activation of a rule. The preconditions are additional tests that help select the correct rule. This slot embodies the reason for applying a particular rule. For instance, the application of the substitution rule might elicit a statement like the following: "Since  $(x+4)$  is repeated in several places in the equation, and  $x$  appears only inside this factor, we substitute  $y=x+4 \dots$  ." The pattern of this rule is:

Mathematical and Planning Knowledge	
Rule:	<i>FACTOR OUT</i>
Pattern:	{ <i>closed expression contains repeated factor</i>
Description:	...
Expectation:	{ <i>Get a factor common with other terms in the equation</i>
Alternative:	<i>REMOVE PARENTHESES</i>
Continuation:	<i>FACTOR OUT</i>
Rule Evaluators	
Performance Measures	
Complexity:	0.5
Length:	0.3
Elegance:	0.8
Error-Proneness:	0.3
Attributes	
Usefulness:	0.9
Importance:	0.8

**Fig. 3.2: Sample Rule in Problem Solving Expertise Module**

PATTERN: { *unknown appears only inside a composite factor, AND composite factor appears more than once in equation*

- ii. The Purpose/Expectation slot contains a description of the result expected from the application of a rule. The difference between purpose and expectation lies in their degree of certainty. The former describes a result that will surely occur, while the latter depicts a desirable outcome of which we have no assurance. A rule may have more than one of these slots; their relevance depends on the equation at hand. In the following example, if we are applying

the operation of removing parentheses on a linear equation, our purpose is to get simple terms. Whereas when applying the same operation on a third order equation, the expectation of getting rid of terms of third degree takes precedence.

RULE: *Remove Parentheses*  
 PURPOSE: *Get simple terms*  
 EXPECTATION: *Eliminate terms of higher degree*

- iii. The slot for Method Description contains a sequence of low level operations which are executed during the application of a rule. For example, Patt's<sup>†</sup> guessing method for solving quadratic equations may be described as follows:

RULE: *Patt's guessing method*  
 PATTERN:  $ax^2 + bx + c$   
 PRECONDITION: *a, b, c are integers*  
 DESCRIPTION: *find two numbers L and M, such that:*  
     *L + M = b, and*  
     *LM = ac = Product-of-Factors(ac)*  
     *method: trial and error of combinations of factors of*  
     *Product-of-Factors(ac)*  
     *rewrite expression as:  $ax^2 + Lx + Mx + c$*   
     *enclose in parentheses two pairs of terms, e.g.,*  
     *(  $ax^2 + Lx$  ) + (  $Mx + c$  )*  
     *factor out common factor for each pair:*  
     *F1 ( composite factor ) + F2 ( composite factor )*  
     *factor out composite factor: ( F1 + F2 )( composite factor )*

- iv. \*The slot for Correctness contains a formal proof of the correctness of the action performed by the rule.
- v. \*The Possible-Continuations slot contains a pointer to other rules that usually

<sup>†</sup> Mr. Patt was a high-school Math teacher in Tel-Aviv, Israel, who devised a very effective method for solving a quadratic equation by means of a product of factors.

\* The starred slots are optional

follow the current one, e.g., removal of parentheses is usually followed by collection of terms. If the set of possible continuations is too large, this slot is omitted.

- vi. \*The Alternatives slot contains a pointer to other rules that could be used instead of the current one. For example, removal of parentheses is an alternative to factoring out common factors. This attribute assists FIGMENT in finding alternative solutions to an equation.

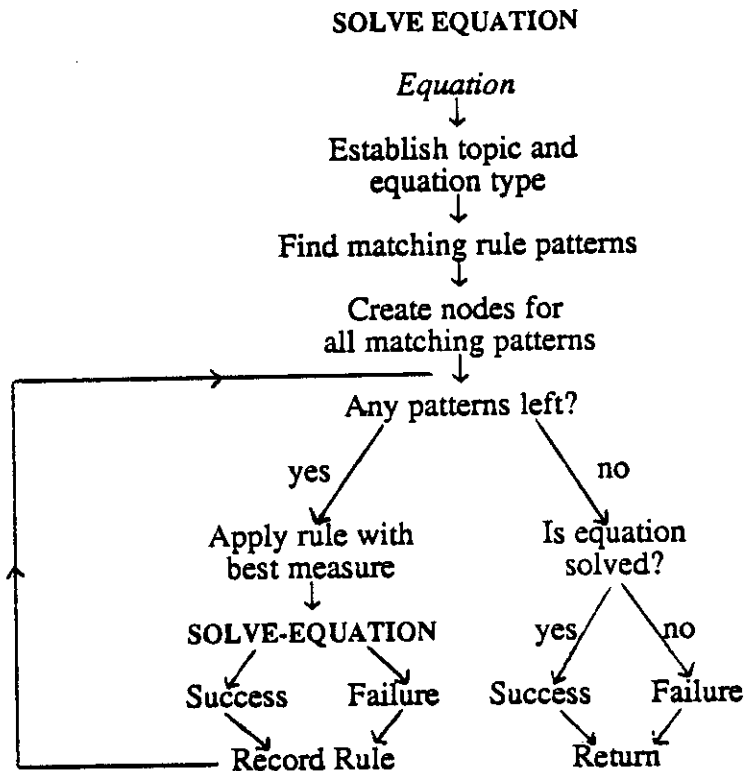
The *Rule Evaluators* component contains several performance measures which provide an assessment of the complexity, length, elegance and error-proneness of a rule. The complexity of a rule represents the difficulty a student has in understanding it and, later on, applying it. The values assigned to these measures represent FIGMENT's approach to algebra, which stresses the choice of methods requiring some thought and insight, as opposed to "brute force" methods that merely entail tedious mechanical manipulations. The former techniques, like substitution and Patt's guessing method, are assigned a high value in elegance and a low one in length and error-proneness; the latter, which include removal of parentheses and collection of terms, usually demand a larger number of calculations and therefore tend to be more error-prone and less elegant. Notice, however, that in general the complexity of the rules preferred by FIGMENT is higher than the complexity of the brute-force rules.

In addition, this component contains an attribute-list similar to the one used for equations with frequently encountered patterns and topics. The attributes of a rule, together with its performance measures, may be used in the generation of a motivational sentence.

### 3.1.2 Problem Solving Knowledge

The Problem Solving Knowledge for algebraic equations is represented by a recursive control structure which operates according to the following backtracking strategy (see Fig. 3.3): when an equation is presented, FIGMENT first tries to establish its topic (e.g., linear, quadratic, etc). Next it searches for the methods whose pattern and preconditions match the given equation, and selects the rule with the best overall measures. The search space may be composed of the rules mentioned in the *continuation* attribute of the previous rule or the rules in the entire knowledge pool in the absence of this attribute. Notice, however, that once the first matching rule is found, its *alternatives* attribute can be used to reduce the search space. The equation which results from applying the selected method is the input for the next call to the problem-solving algorithm. Once FIGMENT reaches a solution or recognizes failure, it will backtrack to the last location in which a choice was made, and generate an alternative solution. The selection of the preferred solution is based on a comparison of the performance measures of all the solution paths attempted.

Since FIGMENT purports to be an effective tutor, its commentaries both explain the solution path chosen and refer to sub-optimal alternatives which may be considered by the student. Therefore, the initial search space for the equation-solving algorithm consists of the rules the student has been taught (see section 3.2). This algorithm is then applied on the entire knowledge pool again, in order to find out whether there is a better way of solving a given equation.



**Fig. 3.3: Problem Solving Knowledge of the PSE Module**

### 3.1.3 Output of the Problem Solving Expertise Module

The final output of the PSE module is a tree-like structure. Its root contains the topic and equation under consideration, its branches consist of the alternatives explored, and each node contains the name of a rule, the objects on which it is applied, and its result (see Fig. 3.4). In order to enable the Comprehension-Processes Module to generate appropriate MTU-codes, additional information like the following may be added:

- i. The pattern of an equation or result, such as *linear canonic form*.



This information is included if an equation (or result) resembles a known equation type or some equations encountered previously.

ii. The expected outcome of a rule application.

Even though this information resides in the Mathematical and Planning component of a rule, the Problem Solving Expert has to select the appropriate expectation or purpose for the current situation. For example, while the expectation slot of the FACTOR OUT rule may have the following default value: "get a factor common to other terms," a particular instantiation of this rule could have the following expectation: "get a factor common to the third term on the right hand side."

iii. The pattern matched to a rule.

This information also exists in the Mathematical and Planning Knowledge component. However, like before, it needs to be added to the output of the Problem Solving Expertise Module in order to present to the student an instantiation of the applied pattern.

iv. The violation or realization of an expectation.

Given an expectation for a particular result, or for a solution path which is similar to a previous one, the PSE module has to recognize when such an expectation is no longer valid, and incorporate this knowledge into the output structure.

v. The certainty of the success of an entire solution path.

This information reflects the confidence of the system in the success of a particular solution alternative. For instance, while the technique of removing

parentheses, collecting terms and applying the quadratic formula will always solve a quadratic equation, the success of Patt's guessing method is rather uncertain.

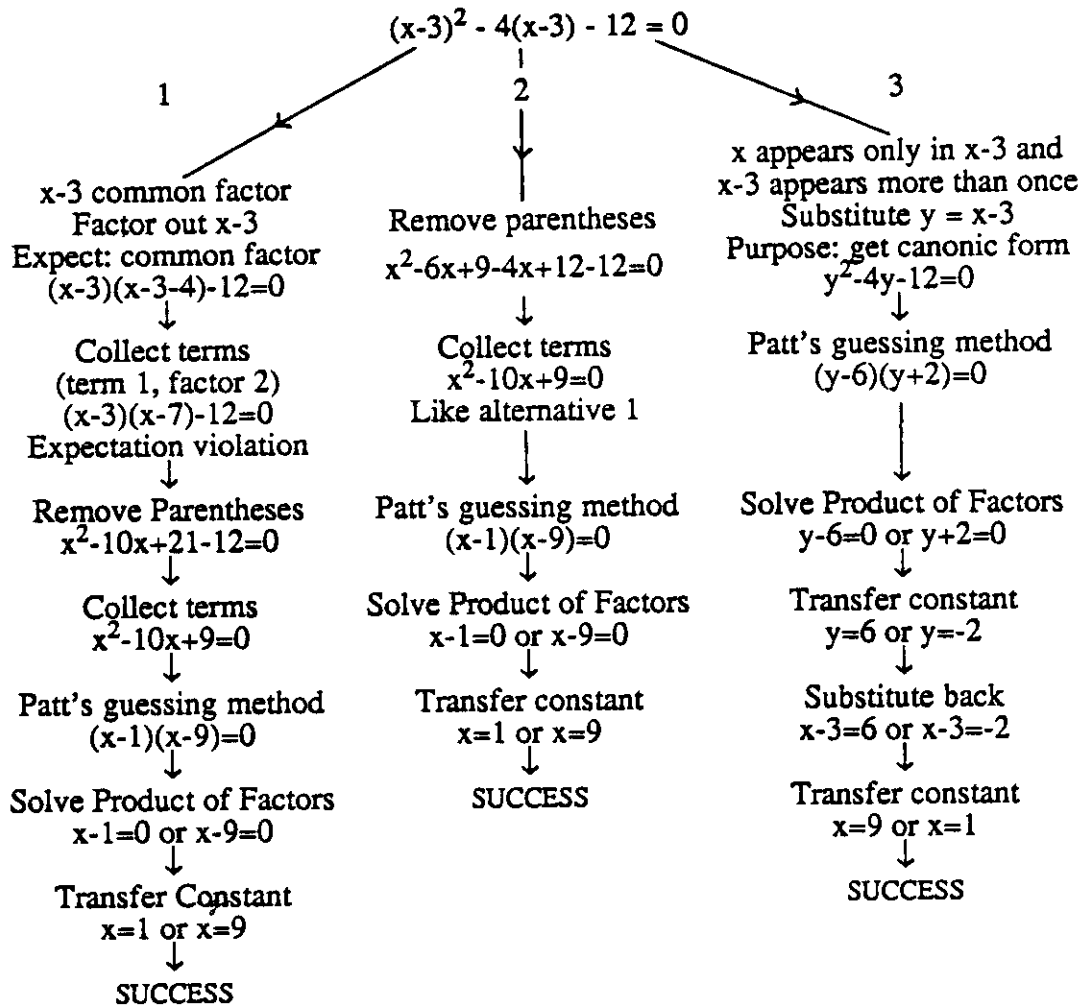


Fig. 3.4: Typical Output of the Problem Solving Expertise Module

In general, due to the simplicity of the domain, once the first algebraic operation is selected the remainder of the solution path is clearly defined, and there is no branching inside a major alternative. Still, the output described in Fig. 3.4 enables the

system to express dynamic relationships such as *sequence* of rule activations, *alternatives* attempted and *violation* or *realization of expectations*. These relationships depend on the solution paths and results arrived at for the equation at hand, and therefore cannot be represented by means of the static information in the Mathematical and Planning Knowledge component.

After the Problem Solving Expert has generated the output structure, the Tutoring Strategist determines which information items from this output structure and the Domain Knowledge will be presented.

### 3.2 Model of the Knowledge of a Student

The Model of the Student's Knowledge is a hypothetical model of the current algebraic knowledge and skill possessed by the student, in which both the long-term and short-term memory of the student are explicitly represented (Collins 1976, Schank and Abelson 1977). The long-term and short-term memory constitute the search space on which the general problem-solving strategies of the student are applied.

The long-term memory of a student is modeled by means of the *overlay* method, i.e., it is considered a subset of the knowledge of the Problem Solving Expert (Goldstein 1981). In order to assess which of the skills of the Problem Solving Expert a student is believed to possess, each item in the Domain Knowledge of the PSE is augmented by its exposure and mastery (see Fig. 3.5).

Exposure — contains the numbers of the equations in which a student was exposed to a particular topic, equation type or rule;

**Mastery** — contains the numbers of the equations in which a student has made proper use of an item of knowledge. For example, if he was able to identify a type of equation, or apply a rule correctly, the number of the equation in which this action was performed is recorded.

In addition, the Model of the Student's Knowledge contains a Motivation slot, which stores information regarding the number of times a student was motivated to attend to a topic, equation type or rule, and also the type of motivation used (see section 4.5 for a detailed description of the different types of motivation).

Finally, the following parameters represent the capabilities of the student.

**Talent** — This is a number between 0 and 1 which evaluates the ability of the student. The most talented student receives a value of 1, the weakest, 0. Initially, a human tutor may input an assessment of this measure, to be modified over a period of time in light of the student's performance. At present, only human input is allowed.

**Diligence** — This is a number between 0 and 1 which measures the industriousness of a student. This evaluation is based on factors like the length of an explanation he is willing to attend to or the number of steps he is prepared to perform when solving an equation.

The information concerning the student's mastery and exposure, together with the evaluation of his talent and diligence are used by the Tutoring Strategist to determine the material to be presented and the depth and length of the explanations (see section 3.3). The Comprehension-Processes Module also uses this information, together with the motivational information, in the generation of MTU-codes.

Mathematical and Planning Knowledge		Knowledge Status		
Rule:	<i>FACTOR OUT</i> <i>closed expression</i> <i>contains repeated factor</i> ... <i>Get a new factor</i> <i>common with other</i> <i>terms in the equation</i> <i>REMOVE PARENTHESES</i> <i>FACTOR OUT</i>	Exposure	Mastery	Motivation
Pattern:		Exposure	Mastery	
Description:		Exposure	Mastery	
Expectation:		Exposure	Mastery	
Alternative:				
Continuation:				
<b>Rule Evaluators</b>				
<b>Performance Measures</b>				
Length:	0.3	Exposure		
Elegance:	0.8	Exposure		
Error-Proneness:	0.3	Exposure		
<b>Attributes</b>				
Usefulness:	0.9	Exposure		
Importance:	0.8	Exposure		

**Fig. 3.5: Sample Rule in Model of the Knowledge of a Student**

The short-term memory of a student contains a list of  $N$  equation instances ( $N$  depending on the student), which he is presumed to remember operationally. Each instance is remembered in as much detail as possible, including its complete path of solution and the alternatives attempted. A model of a student's short-term memory enables a tutor to reference equations similar to the current one, recall alternatives in which the same sequence of operations was applied, etc.

### 3.3 Tutoring Strategies Module

This module receives input from the Problem Solving Expertise module and the Model of the Student's Knowledge, and produces a technical file. It decides on the contents of this file by considering the knowledge-status of the student (i.e., his mastery and exposure), the difficulty and length of the knowledge to be imparted, and certain affects, such as boredom and confusion. For example, in order not to bore a student, a well known fact should not be repeated.

The Tutoring Strategist first determines which of the attempted solution alternatives should be presented to the student and in what order. This decision is made by applying rules like the following:

R1: Do not present more than  $K$  alternatives,

where  $K$  depends on the diligence of the student.

R2: Order the alternatives as follows:

1. First the unsuccessful ones and next the successful ones.
2. In decreasing order of obviousness (i.e., the most obvious or the method practiced lately first).
3. In increasing order of elegance, interest, etc.

Next, the Tutoring Strategist considers each alternative separately, and decides on the operations and results which should be mentioned. The following rules might be applied for this purpose:

R3: Present the first S steps of each alternative,  
where the steps from S+1 to last are well known to the student.

R4: IF a rule is well-known to the student,  
and the result of its application is not significant to the explanation  
THEN omit this result.

Finally, the Tutoring Strategist activates rules like the following, in order to determine which additional technical information should be transmitted:

R5: IF the student's mastery of a topic is medium or less  
THEN mention the topic's name.

R6: IF the student has been exposed to a rule several times and masters it well  
THEN mention the rule (without additional information)  
ELSE IF the student's mastery of a rule is medium or less  
and so is his exposure  
THEN present also the algebraic object on which the rule is applied and  
additional relevant parameters.

R7: IF the student's mastery of a rule's expectation or purpose  
is medium or less  
THEN present this item of knowledge.

### 3.3.1 Output of the Tutoring Strategies Module

The output of the Tutoring Strategies Module consists of a list of technical messages. Each message includes the technical information to be transmitted and *processing* information about the message. For example, an equation technical message has the following format (see Appendix 2 for a detailed description of the different types of technical messages):

TECHNICAL PART: (equation *representation*)

PROCESSING PART:  $\left\{ \begin{array}{l} (\textit{existing-alternatives} \textit{ mentioned-alternatives} \textit{ certainty}) \\ [ (\textit{similar-equations} [\textit{qualifier}]) ] \\ \textit{complexiry} \end{array} \right.$

The *representation* parameter contains the equation under consideration.

The *existing-alternatives* parameter contains the number of alternatives generated by the PSE module, and *mentioned-alternatives* contains the number of alternatives that shall actually be presented; *certainty* has value TRUE if all the alternatives which shall be discussed ensure the solution of the equation, otherwise FALSE.

The *similar-equations* parameter may either contain the numbers of the equations which have a similar pattern or the name of a known pattern (such as linear-canonic-form or quadratic-canonic-form), and the *qualifier* may contain information which restricts this similarity. An example of a qualified similarity statement is: "This equation is similar to the previous one, but it has another term."

Finally, the *complexiry* parameter contains a measure of the difficulty of the current equation.



Fig. 3.6 illustrates the technical part of the output obtained by the Tutoring Strategist from the output of the PSE module presented in Fig. 3.4 (see Appendix 8.2 for a hand-coded listing of this output). In the first alternative, after the expectation for a factor common to the rest of the equation has been violated, the tutor still continues solving the equation by means of the brute-force methods of removing parentheses and collecting terms. If the student is quite proficient in these methods, only their name is mentioned. In addition, the tutor assumes that the student is able to continue solving the equation by himself and terminates the presentation of this solution path. In the second alternative, these brute-force methods are applied from the very beginning and, again, the presentation of the alternative is not completed. Finally, in the third alternative, the tutor decides to spell out the entire solution for a student who has no previous exposure to the substitution method.

From the output of the Tutoring Strategist, FIGMENT proceeds to generate codes which express requirements for different types of MTUs. As stated before, while the codes for Knowledge-Organization and affect-transference MTUs can be directly produced from the output of the Tutoring Strategist Module, the generation of codes for Knowledge-Acquisition and consolatory MTUs entails the simulation of some comprehension processes activated by a student.

TOPIC: quadratic  
EQUATION:  $(x-3)^2 - 4(x-3) - 12 = 0$

(ALTERNATIVE 1)

RULE: factor out x-3 from terms 1 and 2  
PATTERN: x-3 is a factor common to terms 1 and 2  
EXPECTATION: get a factor common with rest of equation  
RESULT:  $(x-3)(x-7) - 12 = 0$

RULE: remove parentheses  
RULE: collect terms  
RESULT:  $x^2 - 10x + 9 = 0$   
CONTINUE

(ALTERNATIVE 2)

RULE: remove parentheses  
RULE: collect terms  
RESULT:  $x^2 - 10x + 9 = 0$   
CONTINUE

(ALTERNATIVE 3)

RULE: substitute  $y = x-3$   
PATTERN: { x appears only in expression x-3  
and x-3 appears more than once  
PURPOSE: get canonic expression  
RESULT:  $y^2 - 4y - 12 = 0$

RULE: quadratic formula  
RESULT:  $y=6$  or  $y=-2$

RULE: substitute back x-3 for y  
RESULT:  $x-3 = 6$  or  $x-3 = -2$

RULE: transfer term  
RESULT:  $x=9$  or  $x=1$   
FINISH

Fig. 3.6: Technical Part of Sample Output of the Tutoring Strategist

## CHAPTER 4

### Generation of Knowledge-Acquisition MTUs: The Comprehension-Processes Module

In order to generate Knowledge-Acquisition MTUs, an effective human tutor generally uses some models of the cognitive-processes triggered in a student upon encountering a technical utterance. These models represent a teacher's perception of the learning habits of a typical student.

In this chapter we examine simplified models of some mental processes presumably activated by a student when reading or listening to a technical utterance. By consulting these models FIGMENT is able to emulate human behaviour for some discourse-generating situations. We prefer these models to a rule-based text-generation system, since the latter contains only an implicit representation of the underlying models. Let us now briefly introduce the mental processes modeled in this work.

i. Building up Motivation and Justification.

This process answers the following mental question: *WHY SHOULD I LISTEN TO THIS UTTERANCE?* A satisfactory answer is necessary to activate the rest of the student's mental activities.

ii. Determining Focus.

The activation of this process attempts to answer the question: *HOW DOES THIS UTTERANCE FIT INTO THE PREVIOUS DISCOURSE?* One of the most valuable

sources of information for building a mental representation of current discourse resides in previously established representations of past discourse. Therefore, a student should be able to recognize whether the tutor is addressing the same topic as before, whether the same equation is being considered, etc.

iii. Determining Type of Relationship.

Once the general context of a technical utterance is clarified, this process is activated in order to determine: *HOW DOES THIS UTTERANCE RELATE TO THE TECHNICAL UTTERANCE CURRENTLY IN FOCUS?* Here a student has to determine whether the forthcoming utterance is a paraphrase of the previous one, an example, a generalization, etc.

iv. Selecting Implementation Mode.

A student may use a given technical utterance to *add* the information contained in it to his knowledge pool, or alternatively, to *verify* or reinforce already existing knowledge. This process is activated to determine: *WHICH OF THESE ACTIONS HAS TO BE PERFORMED?*

v. Applying Mental Resources.

In processing a given technical utterance, a student may either use shallow mental resources (for a simple utterance), deep resources (for a difficult or long utterance), or just apply his default resources. This process is applied in order to decide: *WHICH TYPE OF MENTAL RESOURCES HAS TO BE PREPARED?*

For each technical utterance in the output of the Tutoring Strategies Module, the Comprehension-Processes Module activates these processes to predict the

behaviour of a student in the absence of an MTU. The result of this activation depends both on the information contained in a given technical utterance and on the circumstances surrounding its presentation. If the anticipated behaviour does not agree with the action intended by the tutor, a corrective MTU is generated. For instance, if a student decides to add a particular technical utterance to his knowledge pool, and then discovers that the information contained in this utterance is already there, he might feel disrespect for a tutor who has no recollection of previous discourse. The Comprehension-Processes Module will rectify this situation by producing a verification implementational MTU-code. This MTU-code would eventually have the following English representation: "as you already know," "as I have said before," etc. As another example, if a technical utterance is especially long and/or difficult, a student might fail to process it, in spite of having been provided with appropriate MTUs. In this case, FIGMENT will comfort the student by means of a consolatory MTU like the following: "Do not be concerned, as I will go over this method a few more times."

Our representation of affects differs from Dyer's (1982), in that Dyer provides declarative semantics for affects, while we provide a procedural definition of how a certain affect is reached and, if necessary, avoided. We believe this definition to be sufficient for the generation of MTUs.

The Comprehension-Processes Module examines each technical utterance in the order in which it was produced by the Tutoring Strategist, and generates codes for Knowledge-Acquisition and consolatory MTUs where necessary. The order in which these MTUs are generated for each technical utterance is immaterial, except for the Motivational MTUs which, for reasons of convenience, are produced last.

Knowledge-Organization and affect-transference MTUs may be directly produced from the output of the Tutoring Strategist. However, the requirements for some of these MTUs may be affected by previous technical utterances or Knowledge-Acquisition MTUs, and in turn, the presence of some Knowledge-Organization and affect-transference MTUs may influence the need for other MTUs in later technical utterances. For instance, if a tutor states that the current equation is similar to a previous one (verification implementational MTU), he causes a student to expect both equations to have similar solutions. If, however, their solution paths diverge, this fact has to be advertised by means of an adversative MTU, thereby canceling this expectation. Therefore, for each technical utterance, the Knowledge-Organization and affect-transference MTU-codes are generated immediately after the Knowledge-Acquisition and consolatory MTU-codes have been produced. Only then shall the next technical message be considered.

The order in which a particular configuration of technical and meta-technical utterances appears in the final English representation is independent of the order in which they are generated by the Comprehension-Processes Module. Moreover, some configurations may cause an MTU to become obsolete. In this case, the Sentence Composer eliminates the redundant MTU (see Chapter 6).

#### **4.1 Determining Focus**

A student generally attempts to process a technical utterance in the context provided by preceding discourse. If this is not the context intended by the teacher, the student will feel rather confused. For example, if the tutor completes the solution of a given equation, and wishes to examine another equation, he should provide the student with a focal MTU like "Let us now consider the following equation." Failing to

do so will cause confusion, since the student is expecting the discourse to refer to the previous equation.

A commentary describing the solution of an algebraic equation usually follows the format of the output of the Tutoring Strategist, i.e., first, the equation is presented and its topic is established, then the algebraic rules corresponding to the first approach for solving this equation are considered. This structure of the discourse prompts us to define the *context* in which an utterance is processed, by means of a list which contains the following information:

*(topic equation alternative rule)*

The rightmost non-null element in the list, is defined to be in *high focus*. At the beginning of a tutorial session, the high focus is empty. As the session progresses, the high focus first shifts to a topic and an equation. Next, it may contain an alternative, and finally, a rule. The reader should notice that discussion of a slot in the domain knowledge of an utterance represented in the context, does not produce a shift in focus. For instance, when the pattern or expectations of a rule are being considered, the rule remains in high focus.

Let us illustrate these concepts by means of an example. In the context defined by the list (*linear* "x+3=4" 1 *remove-parentheses*), the tutor is referring to a linear equation, he is examining the first approach applied to solve it, and he is considering the method of removing parentheses. The latter is in high focus. Later on, if the context-list becomes (*linear* "x+3=4" 2  $\emptyset$ ), the discussion still hinges on the same equation, but the tutor is now examining the second alternative. Since the element corresponding to the rule is null, the alternative is in high focus, thus depict-

ing a situation in which the alternative has just been introduced, but the algebraic operations have not been described yet.

A tutor may perform two types of shifts in focus: permanent and temporary.

A Permanent focus shift takes place with the natural progression of the discourse and defines its *permanent context*. If we envision the output of the Tutoring Strategist as a tree, then a permanent focus shift occurs whenever a node is visited in preorder (Knuth 1975). In the following sentences, the expressions in bold-face signal this type of focus shift: “**After removing parentheses**, we collect terms,” “**Let us consider the following linear equation.**”

When a permanent focus shift occurs with respect to an element in position  $i$  of the permanent context, all elements in positions  $1$  to  $i-1$  remain unchanged, whereas all elements in positions  $i+1$  to  $n$  (where  $n$  is the length of the context), are set to null.

A Temporary focus shift (also called digression or interruption, Reichman 1978, 1984 and Grosz & Sidner 1985) occurs when the flow of the discourse is interrupted by an event which has to be addressed by the speaker. This interruption may either be external, like a student asking a question, or internal, e.g., the tutor himself remembering to mention something. After the interruption has been dealt with, the teacher returns to the previous path of the discourse. An internal interruption is signaled by means of MTUs like “*incidentally*” or “*by the way*,” whereas a return to previous discourse is indicated by MTUs like “*anyway*,” “*in any event*,” etc.



A temporary focus shift defines a *temporary context* of the discourse. When a temporary focus shift takes place, the permanent context is temporarily relinquished, and the forthcoming information is processed in the temporary context. When the system signals a return to the permanent context, the temporary context is canceled.

In the following subsections, we discuss the generation of MTU-codes for both types of focus shifts.

#### **4.1.1 Recognizing a Permanent Focus Shift**

In this section we discuss the processes activated to determine the need for a permanent focus-shift MTU. These processes are activated for each technical utterance represented in the permanent context. For a permanent focus shift to become evident, two actions have to be performed: the previous focus has to be closed, and a new focus has to be opened.

##### **Focus close.**

This action terminates the previous discussion. The focus may be implicitly closed through the presentation of a technical utterance. For example, if a tutor says “we remove parentheses,” the focus of a previously applied rule is automatically closed. However, there are cases in which a focus-close MTU may be required. This type of MTU may either be embedded in another utterance, e.g., “Let us now consider the following equation,” or may be represented by an entire sentence like “Let us go on to another topic.”

##### **Focus open.**

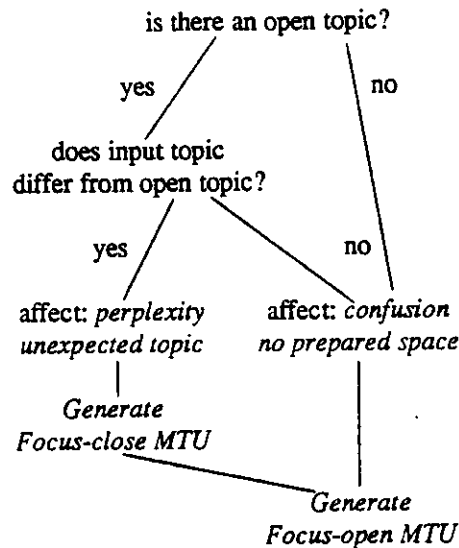
This action introduces a new item, transferring it to high focus. Like before,

this task may be performed by a technical utterance. In the above given example, the sentence "we remove parentheses" not only closes the rule which was previously in focus, but also puts the rule of removing parentheses in high focus. In situations where a technical utterance by itself is unable to open the new focus, a focus-open MTU has to be issued, e.g., a new equation has to be preceded by an introductory MTU like "Let us consider the following equation."

The Comprehension-Processes Module produces codes which express requirements for MTUs that perform these two functions. The generation of these codes is accomplished by consulting simplified models of four focus-recognition processes, one for each type of technical utterance represented in the context. These models are consulted in the following order: first the need for a focal MTU for a topic technical utterance is determined. Next, the focus-recognition process of a student is activated for the equation. Thereafter, the requirements for a focal MTU for the first alternative and for the algebraic operations applied through it are established, and so on.

Fig. 4.1a depicts the focus determination process activated by a student for a topic technical utterance. An *open topic* is the topic a student is expecting, based on the topics of the last few equations.

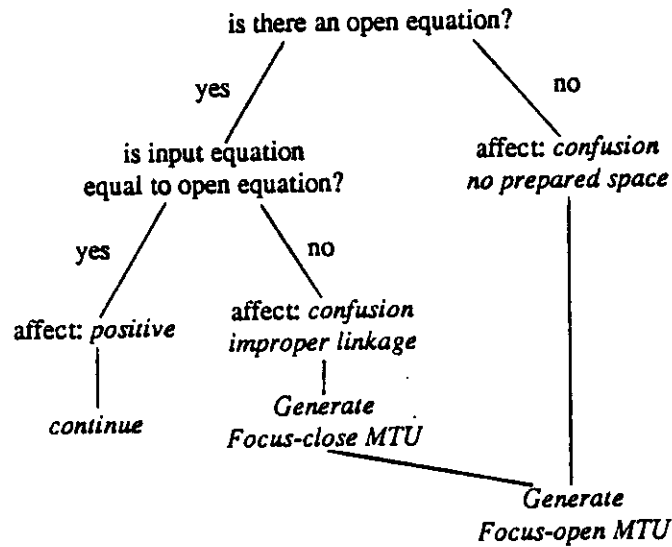
According to the process in Fig. 4.1a, if the student has been practicing equations in different topics lately, he will have no expectations regarding the topic of the next equation, i.e., there will be no open topic, and no focus-closing MTU is required. If, however, there is an open topic, which is not the topic of the forthcoming equation, a student may become perplexed. In this case, a focus-closing MTU-code is generated by the Comprehension-Processes Module. The Sentence Composer then uses this



**Fig. 4.1a: Process for Generating a Permanent Focus-shift MTU for a Topic**

code to produce a sentence like “Let us consider a different topic” or “We shall now examine a quadratic equation” (see section 6.3.3.1). Notice that, while the focus-closing function is performed by the entire first sentence, in the second sentence this activity is performed by the adverb “now,” and the rest of the sentence opens the focus for a new topic. A focus-open MTU should always accompany a topic, in order to prevent a student from experiencing confusion due to failure in preparing a new space.

A diagram such as the one in Fig. 4.1a, is used to describe a mental process presumably activated by a student when trying to understand the presented material. From a strictly pragmatic point of view, the nodes representing the affects could be eliminated. However, we have incorporated them in this dissertation, since they contribute to the plausibility of the models by explicitly presenting the rationale for the generation of an MTU.



**Fig. 4.1b: Process for Generating a Permanent Focus-shift MTU for an Equation**

Upon encountering an equation technical utterance, a process like the one depicted in Fig. 4.1b is activated by the student. In this process, the *open equation* is the equation currently in the permanent context. Notice, however, that if a topic-introducing MTU was issued, there is no open equation, and only a focus-open MTU like the following is required: “Let us consider the following equation” or “Here is the equation.” If a topic technical utterance was not issued then the previous equation is still open and a focus-close MTU has to be generated, yielding a sentence like the following: “We shall now consider the following equation.” This sentence may also be generated by the Sentence-Composer if the following configuration of focal MTU-codes is produced by the last two processes (see section 6.3.3):

<i>topic</i>	close, open
<i>equation</i>	open

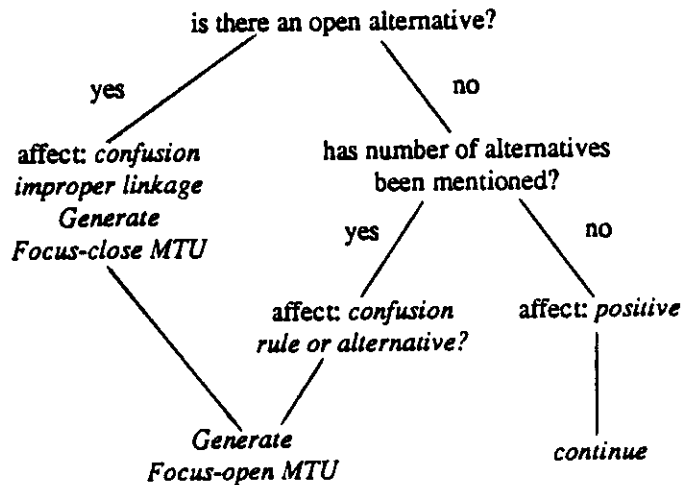
In this case, either of the following introductory texts could be generated:

1. “We shall now consider the following equation: ... . This is a linear equation.”

In this text, the equation technical utterance inherits the focus-close MTU from the topic technical utterance.

2. “Let us now consider the topic of linear equations. Here is an equation.”

Next, when the first alternative technical utterance is encountered, a student activates a focus-recognition process like the one depicted in Fig. 4.1c. This process is repeated for each solution alternative presented by the tutor. Like for equations, an *open alternative* is the alternative currently in the permanent context.



**Fig. 4.1c: Process for Generating a Permanent Focus-shift MTU for an Alternative**

According to this process, if there is an open alternative, then a student shall be confused, since he will not understand how the first operation of the forthcoming

alternative fits in the previous discussion. In this case, the generation of a focus-close and a focus-open MTU is necessary to effect a permanent focus shift. If there is no open alternative, but the number of alternatives to be presented was previously mentioned (see section 4.4.2), then in the absence of a focus-open MTU, a student would experience confusion. Let us illustrate this situation by means of the following examples:

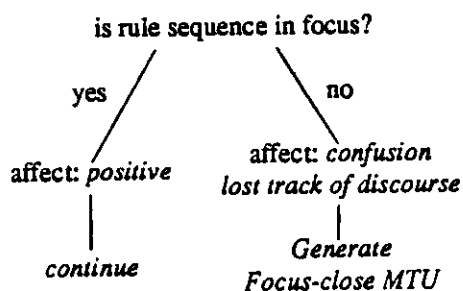
1. “We shall consider three approaches for solving this equation. First, we remove parentheses ... .”
2. “We shall consider three approaches for solving this equation. **The first alternative consists of the following steps:**  
First, we remove parentheses ... .”

In the first example, it is not clear whether the temporal MTU “first” refers to the first alternative or the first algebraic operation, thus causing confusion in the student. This negative affect can be prevented by preceding the description of the first alternative by the focus-opening MTU in boldface, in conjunction with a temporal MTU.

Although this process generates MTU-codes which introduce different avenues for solving a given equation, it does not account for the generation of a hypothetical form of speech. For instance, “If you had factored out  $x-2$  you would have solved the equation. **Instead** you removed parentheses.”

This form is generally reserved for mixed-initiative discourse, in which the alternatives attempted by the student have failed to solve a given equation, or have solved this equation inefficiently, and there is another approach which the student

should have attempted (Davey 1978). In addition, the successful approach has to be rather short, in order to avoid a lengthy discourse in past-perfect tense. Since at present FIGMENT does not analyze students' errors, this form of speech shall not be produced.



**Fig. 4.1d: Process for Generating a Permanent Focus-shift MTU for an Algebraic Rule**

Finally, a student activates the process described in Fig. 4.1d for each algebraic rule. In general, the mere mention of a rule both closes the previous focus and opens a new focus. However, an exception to this situation occurs when, due to a lengthy explanation or digression, the student may have lost track of the operations previously performed. In this case, he needs to be reminded of the algebraic rule which was previously in focus, by generating a focus-closing MTU such as “After dividing by a constant, we collect terms.”

#### 4.1.2 Recognizing a Temporary Focus Shift

There are situations in which the tutor has to temporarily interrupt the flow of discourse, in order to attend to other issues. Once these issues are resolved, he returns to the original discourse. Reichman (1978, 1984) and Grosz & Sidner (1985) classify these situations into the following categories:

i. Digressions.

If something mentioned in one context triggers off an association that leads to an interruption, the speaker has to perform a digression. For example, "This equation can be solved by factoring out  $x-2$ . Incidentally, you could have done so in your test. Anyway, this operation yields a product of factors."

ii. Flashbacks and Filling in missing pieces.

If a speaker forgot or was unable to include important entities in the discourse, he must now interrupt the flow of the discourse, go back and fill in the missing information.

iii. True Interruptions.

An external event causes an interruption if it either disrupts one's line of thought or warrants immediate attention. For example, in a learning environment, this occurs when a student asks a question.

Most human tutors plan a tutorial session in advance. Hence, they know which information they want to transmit about a particular topic or equation type. Furthermore, they have a notion of the types of equations they wish to present, and once an equation is introduced, they know the number of alternatives they wish to discuss, the algebraic operations in each alternative, etc. This characteristic is more pronounced in written or automatically generated discourse, in which a tutor is not likely to forget items of information which are crucial to the understanding of a particular subject. These items are usually delivered while the subject is in high focus. In addition, while presenting a commentary, there are no external interruptions. Therefore, situations which require a temporary focus-transition MTU are restricted to digressions.



As illustrated in the above given example, in order to signal a digression, an MTU like “by the way” or “incidentally” is used. The return to the original discourse is signaled by an MTU like “anyway” or “in any event.”

According to Grosz and Sidner (1985), when an interruption takes place, the permanent context is pushed into a stack. Each subsequent interruption causes the previous context to be pushed into the stack and upon completion of an interruption, this context is popped from the stack. This model is applicable to everyday conversation, since the subjects involved do not demand significant intellectual effort from the speakers. However, in the domain of Tutoring Systems, its applicability is rather limited, since stacking up digressions would be counterproductive to the understanding of the subject. In this situation, if a tutor does perform a digression, he would complete it, and if necessary, transfer to another digression. Otherwise, he would return to the main subject. Thus, instead of stacking subsequent digressions, FIGMENT just replaces the temporary context corresponding to the first digression with the context of the next one. This policy might yield sentences like the following:

“By the way, we can solve linear equations by removing parentheses and collecting terms. I also wanted to mention that we shall not go over a general method for solving third degree equations. Anyway we continue by removing parentheses ... .”

In Fig. 4.2 we present the process used by FIGMENT to determine the need for a temporary focus-shift MTU. The *input context* is the context of the forthcoming technical utterance. If it is null, it signals that this technical utterance should be processed in the permanent context. If no digression is taking place, the temporary context is null, otherwise it contains the context defined by the current digression.

In the current implementation, only general commentaries and statements describing available methods may require a temporary focus-shift MTU. If their accompanying input context is non-null, then a temporary focus shift should be effected. If a general commentary does not have an accompanying context, it should be processed in the permanent context. This policy cannot be applied to method statements, since these may refer only to a topic or an equation with a distinguished pattern. Therefore, if neither of these mathematical entities is in high focus, the context of a method statement is determined by the rightmost non-null element in the permanent context, for which a method statement is relevant. For example, if the permanent context is (*quadratic*  $'x^2+3x-5=0'$  *2 factor-out*), a method statement should be processed in the following context: (*quadratic*  $\emptyset \emptyset \emptyset$ ).

According to the process depicted in Fig. 4.2, if a digression was previously performed (the temporary context is non-null), and the next technical utterance should be processed in the permanent context (the input context is null), then a digression-closing MTU such as "anyway" or "in any event" should be generated. If the input context is not null, then if it is equal to the current temporary context, the digression is continued without further ado; otherwise, a shift in temporary-context has to be signaled by an MTU like "I also wanted to mention that." Finally, if the temporary context is currently null, and the tutor wishes to perform a digression, he should signal his intent by means of a digression-opening MTU like "incidentally" or "by the way."

#### 4.2 Determining Type of Relationship

The relationship between two technical utterances in the same focus is signaled by means of a categorical MTU. Categorical MTUs differ from Knowledge-

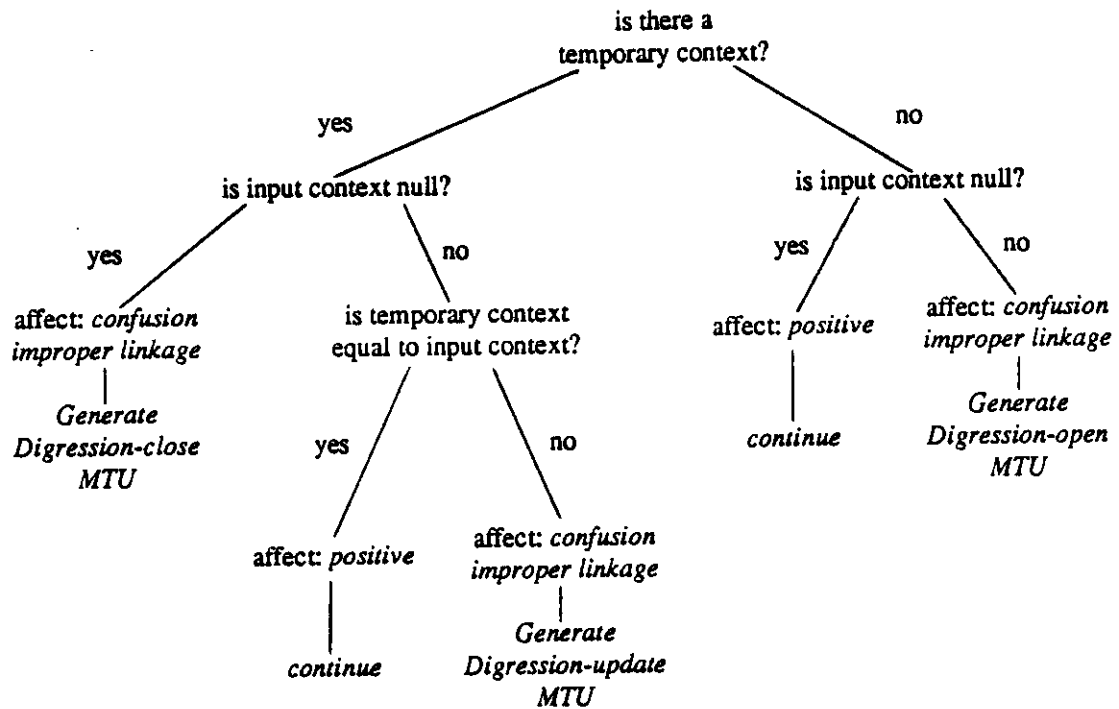


Fig. 4.2: Process for Generating a Temporary Focus-shift MTU

Organization MTUs in that the latter advertise the manner in which two technical utterances should be permanently linked in memory; while the former indicate how a forthcoming technical utterance should be used to update or better understand the previous one. For instance, a paraphrase provides an explicit representation which clarifies a previous implicit representation; an example features an instantiation of a previous abstract statement, etc.

The decision to generate a technical utterance that improves the understanding of a previous utterance is performed by the Tutoring Strategist, and is based both on pedagogical considerations and on affects elicited in the student by its absence. For example, the Tutoring Strategist will decide to issue an example after an abstract

explanation if he feels that its absence may cause a student to experience boredom and frustration. Clearly, in this case, he must be aware of the relationship between the previous and the forthcoming technical utterance, and just needs to transmit this information to the Comprehension-Processes Module.

A categorical relation may also exist between two Knowledge-Acquisition MTUs, or a Knowledge-Acquisition MTU and a technical utterance. For example, "This equation is easy to solve. In fact, we can solve it by ... ." The first sentence consists of an estimational MTU, whereas the second one consists of rule statements. This relationship becomes evident after the Comprehension-Processes Module has completed its operation.

The following examples feature some categorical relationships:

Equivalence — "We take the first term, namely  $3(x-1)$ , ... ."

Generic-Specific — "Some of these techniques, in particular removing parentheses and collecting terms, ... ."

Specific-Generic — "More generally, we can solve linear equations by ... ."

Avowal (contrary to what the current state of the discourse would lead us to expect)  
— "The solution to this equation is rather complex. In fact, we shall not teach it in class."

Summary — "to sum up," "in short."

Exemplification — "for instance," "for example."

The current implementation generates categorical MTUs which signal the presence of an implicit and an explicit description of the same mathematical entity. For example, given the following representation of a factor:

[(1 3 factor) ( $2x^3$ )]

In the absence of an equivalence MTU, the resulting text would be: “the first factor,  $2x^3$ .” FIGMENT evaluates that this description may confuse a student, and concludes that an equivalence MTU is required, yielding the text: “the first factor, namely  $2x^3$ .” The same considerations are applied when generating an equivalence MTU for an introductory statement such as “Let us consider a very interesting topic, namely third degree equations” (see section 6.3.3.1).

### 4.3 Selecting Implementation Mode

A student assimilates a technical utterance by performing one of the following computational activities:

*Addition* of new information to the student’s knowledge pool.

This activity is prepared by means of an implementational MTU like the following: “We shall now introduce the topic of linear equations.”

*Verification* of existing knowledge against an incoming technical utterance.

This activity is triggered by implementational MTUs such as “As I have said before,” “This equation is similar to the previous one.” These MTUs indicate that a particular information item in the student’s knowledge pool should be verified or reinforced by means of the forthcoming technical utterance.

Fig. 4.3a illustrates the model used in selecting an appropriate implementation mode.

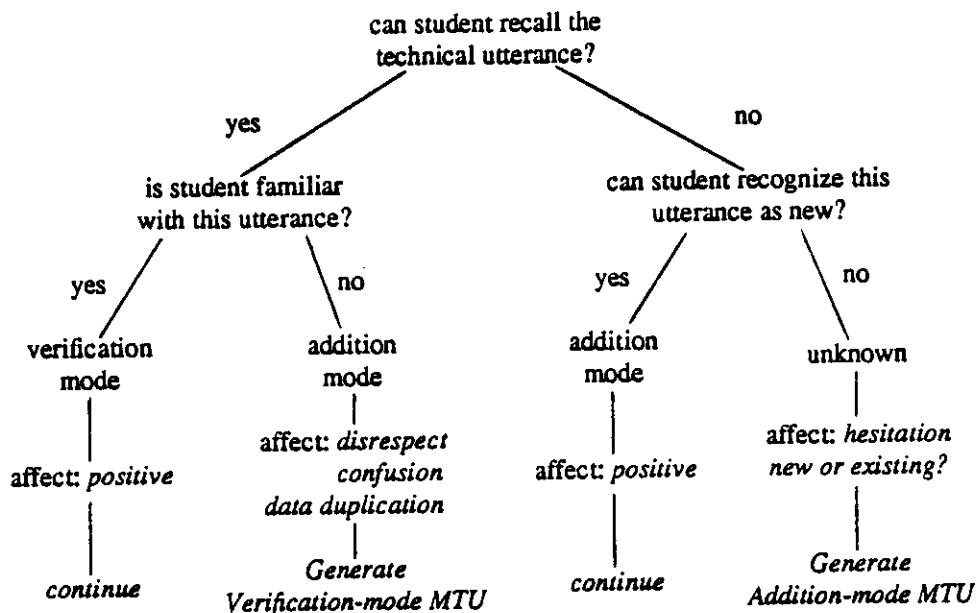


Fig. 4.3a: Process for Generating an Implementational MTU

According to this model, if a student receives a previously encountered but not well-known technical utterance, he will attempt to add it to his knowledge pool (addition mode). Then, upon discovering that the accessed memory location already contains some information, he might experience disrespect for the teacher or confusion. For example, if a tutor repeats an earlier statement, he will elicit disrespect for having no recollection of previous discourse. Likewise, if a student is given an equation accompanied by the statement "This is a linear equation," and this statement is repeated for the next equation, the student may naturally ask: "If this is a linear equation, what was the previous one?" These conflicts can be resolved by generating verification implementational MTUs, yielding statements such as "As I said before,

in general we can solve ... ” or “This is also a linear equation,” respectively. If a student encounters a familiar technical utterance, then even though the absence of a verification MTU may have momentarily put him in addition mode, his well established knowledge transfers him to verification mode. In this case, the presence of an implementational MTU is not only superfluous but even counterproductive, since a student might get bored by the seemingly endless repetition of an implementational MTU. Finally, if a student is presented with a new technical utterance, and is unable to determine whether it was seen previously, he will feel rather confused. To avoid this situation, the teacher has to transfer him first to addition mode by using an implementational MTU like “You have never seen this type of equation before” or “Let us consider a new topic.”

The reader will note that the process of establishing the implementation mode is applied not only to technical utterances, but also to MTUs involving evaluations, yielding text like the following: “As I have said before, this topic is very interesting and important” or “As you already know, this equation is rather simple.” This is because the process depicted in Fig. 4.3a is applicable to any transmitted information.

Another type of implementational MTU, denoted *short-term-implementational* MTU, refers to events in a student’s short-term memory. For instance: “This technique is also easy to apply,” “Like the previous equations, the following exercise can be solved in three different ways” or “This method was also applied in the previous equation.” They are produced by a tutor to exhibit recollection of previous discourse, and differ from implementational MTUs in that, although they prompt the student to recall previous knowledge, this knowledge is not updated.

In order to generate short-term-implementational MTUs, a system would have to search previously generated extended messages for MTU-codes common with the current MTUs, and apply rhetorical rules to determine the need for short-term-implementational MTUs. Since the current version of FIGMENT maintains a partial model of a student's short-term memory (see section 3.2), we have not implemented a procedure for generating these MTUs. They are issued only in one particular case, i.e., they accompany MTUs which motivate a student to attend to a solution alternative, yielding text like the following: "Another alternative which enables us to practice this method consists of the following steps."

Let us now consider the manner in which we test the conditions required by the process of Fig. 4.3a.

*Can the student recall the technical utterance?*

The current system does not contain an accurate model of the processes by which a student remembers or forgets an information item. Our model simply assumes that if a student was exposed to a particular technical utterance, he should be able to recall it.

*Is the student familiar with the utterance?*

The answer to this question is arrived at by means of the following rule:

IF NUMBER-OF-UTTERANCES > RECOGNITION-THRESHOLD  
THEN ASSERT *student is familiar with utterance*

In order to perform this test, the system stores a RECOGNITION-THRESHOLD for each type of technical utterance in the domain knowledge, namely topics, equations, algebraic rules and statements. The different thresholds are required due to the different degrees of difficulty in recognizing a



particular utterance.

It is natural to think that a student's familiarity with a technical utterance depends also on his talent, i.e., a more talented student should need fewer repetitions of a particular utterance than a mediocre student. In practice, however, this consideration turns out to be unnecessary, since the number of times an implementational MTU can be issued without causing a student to be bored is quite limited.

*Can the student recognize this utterance as new?*

The system answers this question by performing the following test:

IF TALENT-OF-STUDENT > TALENT-THRESHOLD  
THEN ASSERT *student can recognize new utterance*

Our basic paradigm is that a talented student will recognize that a given utterance is new, without being explicitly informed of this fact. A mediocre student, however, might think that he has forgotten a previously mentioned utterance, and may not know whether to continue searching for it. A student's reaction also depends on the type of the technical utterance under consideration. In a tutorial setting, if a student is unfamiliar with an equation with a not-distinguished pattern or a statement, he assumes that it is new, unless otherwise stated. Moreover, if a student were advised of every new equation and statement, the resulting text would be awkward and repetitious. This is not so for topics, equations with distinguished patterns and rules, since their recognition requires some ability. Hence, in order to perform this test, FIGMENT stores a TALENT-THRESHOLD for each of these types of technical utterances. The threshold for equations with not-distinguished patterns and for statements

is quite low, while the threshold for topics, equations with significant patterns and rules is higher.

After the process depicted in Fig. 4.3a has determined that a verification implementational MTU has to be generated, the actual code (and subsequent text) depends on the circumstances and the type of utterance under consideration. For example, if the topic of the previous equation is equal to the topic of the current equation, then the equation should be introduced by means of a sentence like "Let us continue with a linear equation" or "This is also a linear equation." If the topics differ, and more than one equation shall be presented, then a sentence like the following is issued: "We shall now return to the topic of quadratic equations." When an equation technical utterance is being considered, then a student is transferred to verification mode by means of a sentence such as "This equation is similar to the last one" or "We have studied this type of equation before," depending on whether the equation is of an established pattern. These MTUs direct the student towards updating a useful pattern and inhibit the creation of irrelevant patterns. Finally, a statement which requires an implementational MTU, would be preceded by text like the following: "As you already know," "As I have said before," etc.

The process depicted in Fig. 4.3a applies to algebraic rules in a general way. However, unlike topics, equations and statements, a rule can be applied on various objects, in different locations, etc. In order to distinguish between the different situations pertaining to the application of rules, a more detailed process is necessary (see Fig. 4.3b). The right hand branch of the discrimination net in Fig. 4.3b is equal to the right hand branch of the process in Fig. 4.3a. The left hand branch, however, demands careful consideration.

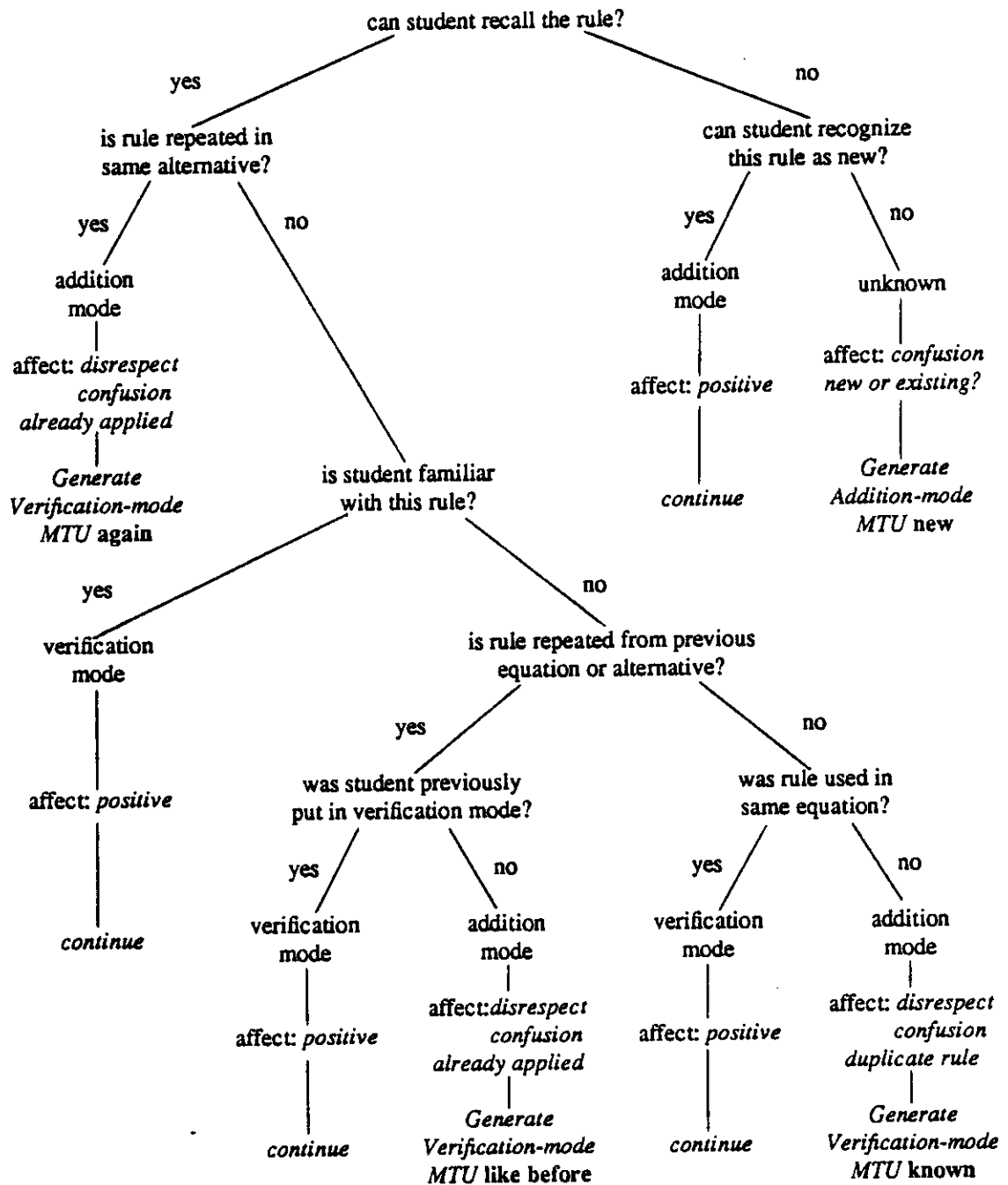


Fig. 4.3b: Process for Generating an Implementational MTU for an Algebraic Rule

In order to determine the code (and subsequent text) of the implementational MTU to be generated, the process depicted in Fig. 4.3b distinguishes between various degrees of similarity among different applications of the same rule.

Repetition in the same alternative.

If an algebraic rule is used more than once in the solution of an equation, the tutor states that it is applied to the same object or using the same instrument, and the rest of the arguments of the rule are not being stated, then an MTU like “again” or “once more” has to be used. For example, given the equation:

$$x^3 - 5x^2 + (x+1)(x-1) + 1 = 0$$

The following text would be generated:

First, we factor out  $x^2$  from the first and second terms, yielding:

$$x^2(x-5) + (x+1)(x-1) + 1 = 0$$

Next, we apply the formula  $(a+b)(a-b)=a^2-b^2$  to the second term

and collect terms, which results in:

$$x^2(x-5) + x^2 = 0$$

Thereafter, we factor out  $x^2$  once more, with the following result:

$$x^2(x-4) = 0$$

⋮

Omission of this MTU may cause the naive student to think that it is odd to perform the same operation on the same object twice, and that one of these operations is probably a mistake. This MTU services also the sophisti-

cated student who may feel disrespect for a tutor who does not seem to recall having just performed the same operation. The reader should notice that, unlike the rest of the technical utterances, this type of MTU is generated regardless of the student's ability to recognize the rule in question.

**Repetition in a previous alternative or equation.**

If the tutor has already pointed out that the current equation resembles a previous one, he has transferred the student to verification mode, causing him to expect the sequence of rules applied to solve the referenced equation, and so eliminating the need for implementational MTUs. If, however, the similarity between the equations was not previously mentioned, the repeated application of an unfamiliar rule will require the generation of an MTU such as "We now remove parentheses, like in the last equation."

**Previous usage.**

This type of similarity refers to an algebraic rule which has been previously applied, however, not under circumstances which resemble the current application. In this case, if a student is not familiar with the rule, and it was not used in the same equation, an implementational MTU like "We have encountered this technique before" has to be issued. If this rule was already used in the same equation, an implementational MTU of this type would be superfluous and repetitious.

#### **4.4 Applying Mental Resources**

In general, a student expects a technical utterance to be of average length and difficulty. This prompts him to prepare default mental resources for processing it.

There are, however, technical utterances which do not conform to the student's expectations, i.e., they may be rather lengthy or difficult or, alternatively, they could be extremely simple. In these cases, as shown in Figs. 4.3 and 4.4, the discordance between the student's expectations and reality might cause a waste of useful resources.

FIGMENT is able to predict most of these situations by assessing the knowledge status and quality of a student against the attributes of the problem under consideration. This evaluation allows him to advance conjectures regarding technical utterances which the student will most likely fail to process, and to generate warnings such as "The following equation is quite simple" and "We shall devote most of the session to quadratic equations." Moreover, when failure is anticipated despite having generated adequate estimational MTUs, a consolatory MTU has to be produced, e.g., "I know these are a lot of equations, but it is necessary." In the following subsection we shall consider the model used by FIGMENT to generate complexity-related MTUs. Thereafter we shall discuss the model used to produce length-related MTUs.

#### 4.4.1 Preparing Computational Power

To assist a student in selecting the level of computational power required to process a given technical utterance, a tutor generates a *complexity-related* MTU. Fig. 4.4 depicts a model of the process activated by the system to generate this type of MTUs. Before we discuss this process in detail, let us consider a working definition of the term *complexity* for the technical utterances under consideration:

Topic — The complexity of a topic is a number between 0 and 1, defined as a combination of the complexity of its definitions, theorems, algebraic rules and equa-

tions;

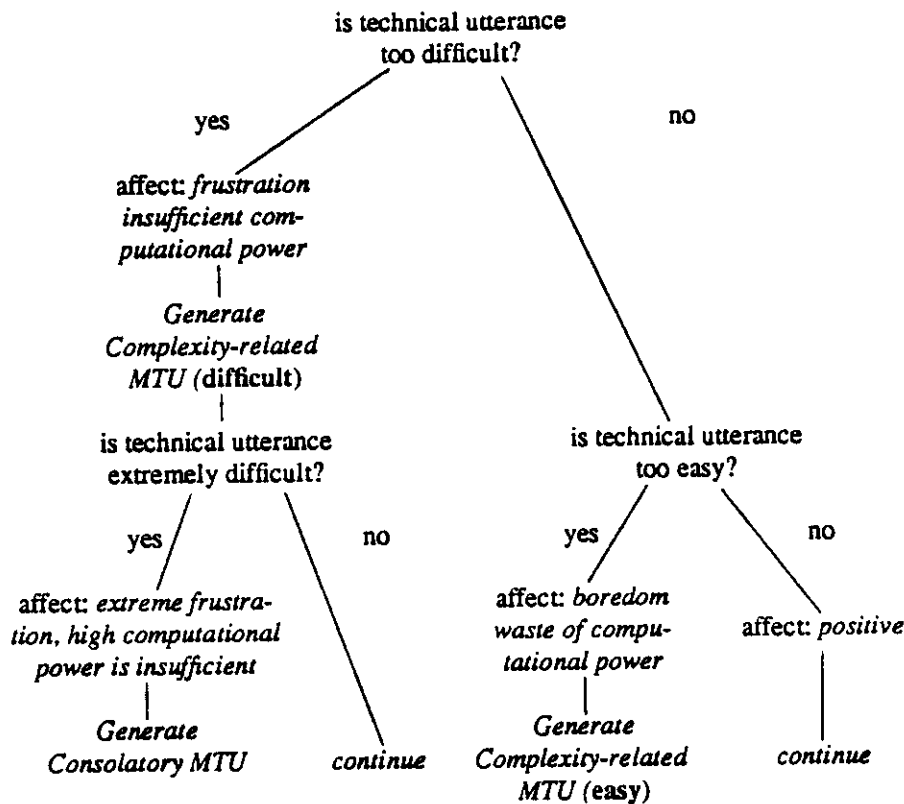
**Equation** — The complexity of an equation, is a number between 0 and 1 which measures the difficulty a student of average talent will experience in arriving at its solution;

**Alternative** — The complexity of a solution alternative is the sum of the complexity measures of its rules; and

**Definitions, Theorems, Rules and Statements** — Their complexity is also a number between 0 and 1 which measures the difficulty a student of average talent has in understanding and using them.

In the current implementation the process depicted in Fig. 4.4 is applied to topics, equations and rules, and not to entire solution alternatives or statements. This is due to the fact that the complexity of a description statement is directly related to the complexity of the rule which it describes, and pattern, expectation and method statements are factual statements which do not require a complexity-related MTU; a correctness proof might require a complexity-related MTU, but at this point it is not being generated. A solution alternative does not require a complexity-related MTU, since the complexity of an equation already conveys the difficulty experienced by a student in solving it, hence it would be either repetitious or contradictory to advertise the difficulty of any of its solution alternatives.

In the process illustrated in Fig. 4.4 the system first evaluates whether a student will experience difficulty in processing a particular technical utterance. It checks whether the default resources are sufficient, and if not it advises the student of the utterance's complexity by means of a statement like "This equation is hard to solve"



**Fig. 4.4: Process for Generating a Complexity-related MTU.**

or “This technique is very difficult to apply.” These complexity-related MTUs will prompt a student to prepare high computational power (i.e., increase his concentration). Clearly, the presence of a complexity-related MTU does not ensure success in processing a technical utterance; nevertheless, it increases the student’s chances of succeeding, and even in the event of failure, he might find comfort in the fact that he failed while trying to perform a difficult task.

Next, the system checks whether there is a great disparity between the capabilities of the student and the difficulty of the utterance. If so, even though the student is being provided with an adequate preparatory MTU, the deep mental activity will not



suffice to process the technical utterance under consideration, causing frustration. The Comprehension-Processes Module assumes that the Tutoring Strategist has already tried to optimize the difficulty of the utterance. Therefore, it only tries to alleviate negative affects by adding a consolatory MTU such as "Don't worry, we shall go over this equation several times."

Alternatively, if the technical utterance is very simple, the student will certainly succeed in processing it, but he might become bored and inattentive. In this case, the tutor should warn him to prepare low computational power, using MTUs such as "clearly," "obviously," "this equation is rather straightforward," etc.

The difficulty of the information the student can comfortably digest depends on the complexity inherent in the technical utterance, the talent of the student and his previous mastery of this utterance. A talented student, who has exhibited proficiency in solving a given type of equation, will not require a complexity-related MTU to advise him of its difficulty. The following rule is used by FIGMENT to encode these considerations:

```
IF { RELATIVE-UTTERANCE-COMPLEXITY — TALENT-OF-THE-STUDENT } >
    UPPER-COMPLEXITY-THRESHOLD
AND TECHNICAL-UTTERANCE-COMPLEXITY ≥ 0.2
THEN ASSERT technical utterance is too difficult
```

Where:

$$RELATIVE-UTTERANCE-COMPLEXITY = \frac{COMPLEXITY-OF-TECHNICAL-UTTERANCE}{\sqrt{RELATIVE-EXPERTISE + 0.25}}$$

And RELATIVE-EXPERTISE is determined by the following ratio:

$$RELATIVE-EXPERTISE = \frac{NUMBER-OF-UTTERANCES-MASTERED}{NUMBER-OF-UTTERANCES-EXPOSED-TO}$$

According to this measure, a student is advised of the difficulty of a technical utterance, only if its RELATIVE COMPLEXITY exceeds his TALENT by more than a certain THRESHOLD. Still, a complexity-related MTU shall not be generated for an utterance whose complexity measure is below 0.2, regardless of the student's qualifications.

In this formula, the student's RELATIVE-EXPERTISE is calculated over the last N technical utterances of the type of the utterance in question, thus discounting poor performances in the past. The RELATIVE-EXPERTISE has been adjusted by adding to it 0.25, thereby considering a previous expertise of 75% an acceptable score. The square root of the denominator serves to moderate the weight of the RELATIVE-EXPERTISE compared to the COMPLEXITY measure.

Similarly, the need for a consolatory MTU is ascertained by means of the following rule:

```
IF { RELATIVE-UTTERANCE-COMPLEXITY — TALENT-OF-THE-STUDENT } >
    EXTREME-COMPLEXITY-THRESHOLD
    AND TECHNICAL-UTTERANCE-COMPLEXITY ≥ 0.4
    THEN ASSERT technical utterance is extremely difficult
```

Finally, FIGMENT determines that a student can get by with only low computational power, by means of the following rule:

```
IF { RELATIVE-UTTERANCE-COMPLEXITY — TALENT-OF-THE-STUDENT } <
    LOWER-COMPLEXITY-THRESHOLD
    AND TECHNICAL-UTTERANCE-COMPLEXITY ≤ 0.8
    THEN ASSERT technical utterance is too easy
```

As before, the system will not advertise the ease of a technical utterance if its complexity measure is greater than 0.8.

The preceding rules convey qualitative relationships between the need for a complexity-related MTU and a student's talent and relative-expertise. The numerical parameters in these rules were adjusted to yield satisfactory outputs in a large number of repeated tests. (For numerical results of the application of these rules with different values for talent, expertise and complexity, see Appendix 6.1).

We found that the need to issue a complexity-related and a consolatory MTU should depend not only on the utterance's attributes and the student's qualifications, but also on the state of the discourse. Therefore, before generating MTUs of these types, the following rules are applied:

- i. If the complexity of a rule was already stated in the current equation, it should not be repeated.
- ii. If a complexity-related MTU is being repeated, it should be preceded by an implementational MTU, yielding text like the following: "As I said before, this technique can be easily applied."
- iii. If a complexity-related MTU was already stated a certain number of times (say 3), then it should not be repeated.
- iv. A consolatory MTU may be stated only the first time a technical utterance is presented or for equations without a distinguished pattern.

#### **4.4.2 Preparing Processing Time**

In general, a student is quite prepared to devote a certain amount of time to process a technical utterance. He would probably be quite content if the utterance required less processing time than anticipated. However, should the forthcoming

utterance need longer processing time than expected, a loss of attention might ensue. A tutor should be able to predict these situations and prevent the ensuing negative affects by generating a *length-related* MTU. Figure 4.4 illustrates the process activated by the system to generate this type of MTU. However, before we discuss this process in detail, let us define the *length* of the technical utterances under consideration:

**Topic** — The length of a topic is defined by an estimate of the number of consecutive equations a tutor predicts he is about to present. In general, a tutor will not anticipate the exact number of equations that will be discussed, but a range of equations (see Appendix 2). For example, the following length-related MTUs are often encountered during a tutorial session: “Let us now consider a few linear equations,” “We shall devote the rest of this session to third degree equations,” etc. If no prediction is made, the student assumes that, for the moment, one equation in a given topic will be discussed.

**Equation** — We define the length of an equation as the number of solution alternatives which will be examined during a commentary. Clearly, this number may differ from the number of existing solution paths. The generated length-related MTUs distinguish between the number of mentioned alternatives and the number of existing alternatives by means of expressions like the following: “We shall discuss two approaches for solving this equation” and “There are two approaches for solving this equation,” respectively.

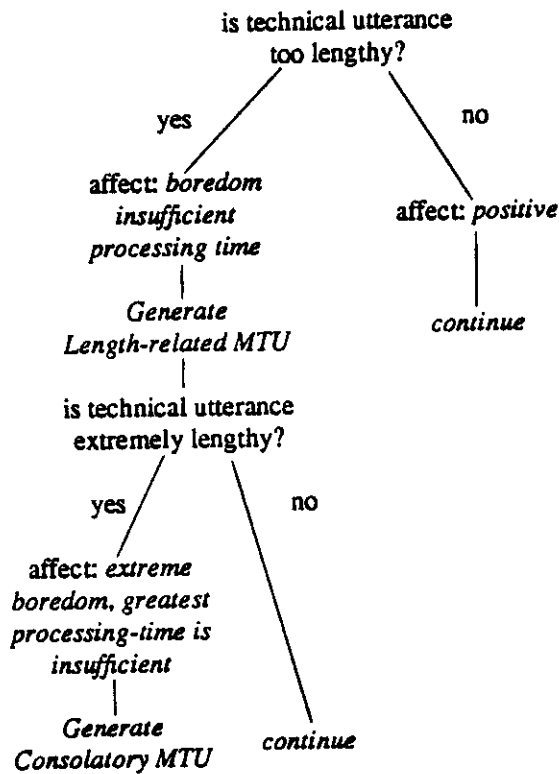
**Alternative** — The length of an alternative is defined as the number of algebraic rules that solve the equation. When discussing the solution of an equation, a teacher may omit some of its steps. Thus the number of steps actually performed may

differ from the number of operations discussed. However, unlike the length of an equation, the length of an alternative is defined as the number of existing steps. This is because the alternatives which are not mentioned do not have to be processed by the student, while the omitted rules still have to be processed, though possibly in a cursory manner.

**Rule** — The length of a rule is a number between 0 and 1, which is a measure of the amount of computations performed for one application of the rule on a minimal number of elements. For example, the length of the application of the quadratic formula is 0.9, and the length of removing parentheses is 0.2. However, while the former value is fairly constant, the latter depends on the number of terms which have to be multiplied. The length-related MTUs distinguish between the length inherent in a rule and the length resulting from numerous applications of a (probably short) rule by means of statements like “This technique entails many computations” and “This technique entails many computations, in this case,” respectively.

**Statement** — We define the length of a statement as a number between 0 and 1 which represents a normalized evaluation of the number of clauses in its English representation. In the realm of Tutoring Algebra, most statements are adequately represented by a few English clauses. However, a correctness proof or a detailed description of the manner in which a rule is applied could be quite lengthy.

According to the process depicted in Fig. 4.5a, a length-related MTU is generated if a technical utterance is judged too lengthy. For example, “This technique entails several calculations.” Although this preparation does not ensure success in



**Fig. 4.5a: Process for Generating a Length-related MTU**

processing the utterance at hand, it is our contention that the preparatory MTU increases the student's chances of coping successfully with it. If the teacher anticipates that even this preparation is insufficient to process a technical utterance, the length-related MTU will be followed by a consolatory MTU like the following: "This alternative involves many steps, but it has to be examined."

We have not found it necessary to generate a length-related MTU for a technical utterance which is shorter than the default expectation. Still, there are special circumstances in which such warnings are in order. For example, if a student is tired and bored, but the teacher wishes to continue discussing the material, he might say "Let

us solve just one more equation” or “The solution to this equation is really short.” Unlike the MTUs presented previously, these MTUs are not issued to prevent negative affects, but to alleviate existing ones. Another scenario which may require MTUs of this type occurs when a student falsely expects the forthcoming utterance to be of a length identical to that of the preceding one. In this case, a length-related MTU like the following, probably accompanied by an adversative MTU, might be generated: “Unlike the last equation, there is only one way to solve this exercise.” Finally, in addition to the MTUs generated by the process depicted in Fig. 4.5a, the Comprehension-Processes Module may issue an MTU like “There is only one way to solve this equation,” in order to prevent a student from fruitlessly searching for alternative solutions. This type of MTU refers only to the number of existing alternatives, as opposed to the number of alternatives which will be discussed, and its generation is independent of the diligence of the student.

The processing time a student can comfortably devote to a technical utterance depends both on its length and on the diligence of the student. Therefore, the system evaluates the need for a length-related MTU by means of the following rule:

IF RELATIVE-UTTERANCE-LENGTH > UPPER-LENGTH-THRESHOLD  
 THEN ASSERT *technical utterance is too lengthy*

The UPPER-LENGTH-THRESHOLD is defined as the maximum utterance length which can be comfortably processed by a student whose diligence is 1, and the RELATIVE-UTTERANCE-LENGTH is defined as:

$$RELATIVE-UTTERANCE-LENGTH = \frac{LENGTH-OF-THE-TECHNICAL-UTTERANCE}{\sqrt{\max(DILIGENCE-OF-THE-STUDENT, 0.2)}}$$

To prevent the generation of a length-related MTU for a short utterance, the denominator in the formula for the calculation of the RELATIVE-LENGTH is adjusted by a lower bound, namely 0.2.

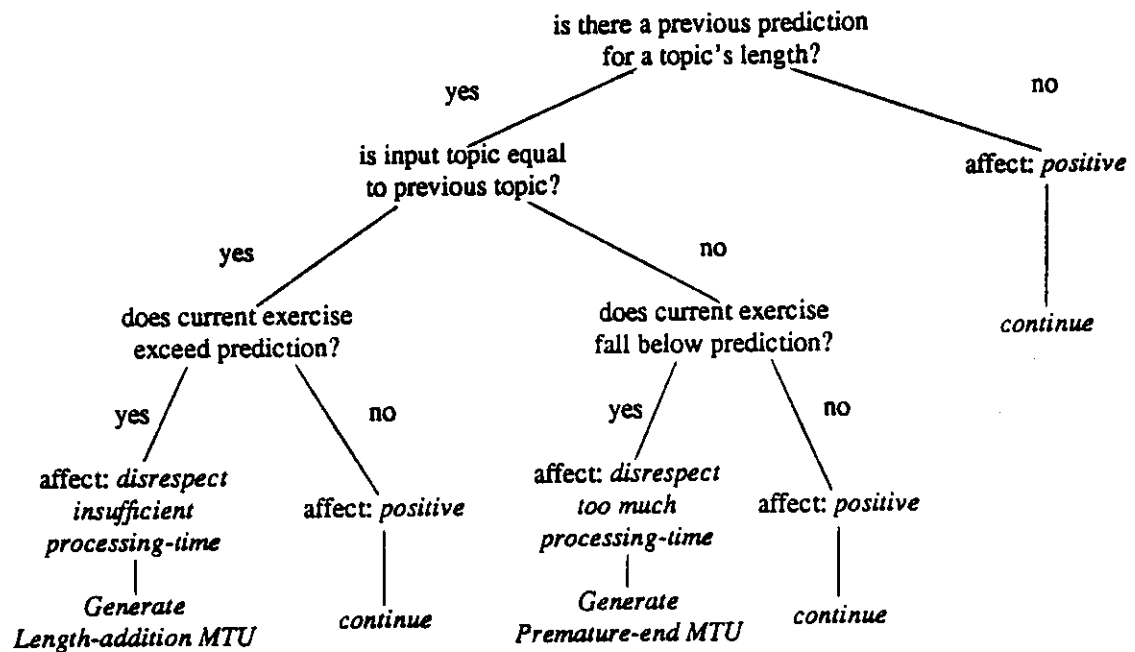
Likewise, FIGMENT determines the need for a consolatory MTU by the following rule:

IF RELATIVE-UTTERANCE-LENGTH > EXTREME-LENGTH-THRESHOLD  
THEN ASSERT *technical utterance is extremely lengthy*

The process of determining the need for a length-related MTU is basically the same for all types of technical utterances. However, whereas the Tutoring Strategist has firmly established the length of equations, alternatives and rules, a prediction regarding the length of a topic may remain unfulfilled, due to new information on the student's performance. For instance, if a student's performance is worse than expected, he will have to practice additional equations, or alternatively, if a student's performance has improved dramatically, he may require less than the anticipated number of equations to achieve proficiency. Therefore, prior to activating the process depicted in Fig. 4.5a, the system needs to ascertain whether a prediction regarding the length of the previous topic has been fulfilled, and if not, generate appropriate MTUs. This task is performed by the process described in Fig. 4.5b.

According to this process, if the number of exercises in the current topic exceeds the anticipated number of exercises, a student will doubt the tutor's ability to recall his prediction. In this case, FIGMENT has to generate a length-related MTU like the following: "Let us examine another quadratic equation." The reader will recall that the English representation for this type of MTU is equal to the representation of a verification implementational MTU. Therefore, a requirement for both types





**Fig. 4.5b: Process for Generating a Length-related MTU for a Previous Topic-length Prediction**

of MTUs shall converge to one surface representation (see section 6.3.3.1). Alternatively, if a change in topic is effected before the predicted number of equations is discussed, a student should expect some explanation regarding this turn of events. In this case, FIGMENT might generate the following MTU: "I know I said we would solve more linear equations, but I feel we have done enough."

When presenting a length-related MTU for an alternative, if a consolatory MTU is also required, then FIGMENT checks whether the number of steps which shall be mentioned is acceptable to the student. In this case, the consolatory MTU would be composed of the number of steps to be mentioned, yielding text such as "The following solution alternative requires many steps, however we shall examine only a few of them." Otherwise, a consolatory MTU like the ones presented for

equations and topics is issued.

The need to issue a length-related and/or consolatory MTU depends on the status of the discourse. Therefore, prior to issuing MTUs of these types, the following rules are applied:

- i. If the length inherent in a rule was already stated in the current equation, it should not be repeated.
- ii. If a length-related MTU is being repeated, it should be preceded by an implementational MTU, yielding text like the following: "As I said before, this method entails many calculations."
- iii. If a previously stated length does not match the current length, then the latter should be omitted.
- iv. If a length-related MTU was already stated a certain number of times (say 3), then it should not be repeated.
- v. A consolatory MTU may be stated only for topics, equations without distinguished patterns and alternatives, and the first time an equation with a distinguished pattern, a rule or a statement is presented.

#### **4.5 Building up Motivation and Justification**

An algebra student typically exhibits the goals of solving the given equations and mastering the subject matter. Ideally, the fulfillment of the second goal is a precondition for the fulfillment of the first one. In addition, a student may have social goals like earning the respect of his peers or gaining the approval of the teacher.

Finally, while studying the material, a student has the more immediate goal of remaining interested and, if possible, amused.

To motivate a student, a teacher may remind him of the status of his goals and the actions he has to perform to achieve them. However, a tutor is not expected to say "You need to master this subject so that Sally will admire you." Instead, he will attempt to motivate a student through his knowledge-related goals. This task is performed by using a variety of MTUs which reflect the tutor's perception of the knowledge-status of the student. For instance:

1. "We shall now consider a topic, namely quadratic equations, which we have not seen for a while."

This motivation is issued to prompt a student to practice a topic which he may be forgetting.

2. "This alternative serves to introduce the very important and interesting method of factoring out common factors."

A tutor uses this motivation to awaken interest in a new item of knowledge.

3. "This type of equation has been practiced a lot, but it still demands some more practice."

This motivation is generated to encourage a, probably tired, student to continue practicing a subject in which he lacks proficiency.

The design of FIGMENT captures important aspects of the motivation-generation process, enabling it to produce these and other MTUs.

We recognize two types of motivations related to the student's goal of mastering the subject matter:

**Knowledge Preservation** — If a certain item of information has not been encountered for some time, and the tutor suspects that the student's skills might have deteriorated, he could motivate the student by means of a *knowledge preservation* motivation, yielding a sentence such as "This equation enables us to practice a technique, which we have not encountered for a while"; and

**Increment Knowledge** — If a technical utterance has been practiced recently, we can safely assume that the expertise of the student will only increase with additional practice. In this case, if the performance of a student leaves something to be desired, a tutor can use an *increment knowledge* motivation, producing a sentence like the following: "Let us continue with the following type of equation, which demands some more practice."

To keep a student interested, a tutor may *highlight the attributes* of an information item; for instance, "This topic is very interesting and challenging." This type of motivation is generally used when introducing a new technical utterance. However, it sometimes accompanies a knowledge-status related motivation statement, yielding a sentence like the following: "Let us consider a very interesting topic, namely third degree equations, which requires additional practice." A similar type of motivation highlights the relative attributes of a method or a topic, for example, "This method is more elegant and efficient than removing parentheses and collecting terms."

As stated in section 3.3, FIGMENT occasionally presents solution alternatives which the student may have considered. These may be motivated by the following

sentence: "Another alternative you might have thought of consists of the following steps." This motivation satisfies the student's goal of mastering the subject matter, as well as a social goal, by finding out the tutor's opinion about him.

Finally, if none of these motivation types is applicable, a tutor could motivate the student by appealing to his social goals, i.e., gaining the teacher's approval. This motivation is represented by a sentence such as "I would like you to solve the following equation."

Before we consider in detail the motivation process used by the tutor, we would like to discuss its relation to the context hierarchy (see Fig. 4.6).

**Inheritance** — If we motivate a student to attend to the solution of a given equation through the equation's topic or its type, we may assume that the student has acquired the goal of solving it, and that this goal remains active as long as the equation is not solved. Therefore, all unsuccessful solution alternatives and the first successful alternative inherit this motivation, and the student does not have to be motivated separately for them (recall that the Tutoring Strategist presents first the unsuccessful alternatives and then the successful ones). However, the student still has to be motivated to attend to additional alternatives.

Similarly, the need for a knowledge-status related or social motivation may also be inherited, i.e., if a student lacks skill in a particular topic, his lack of expertise is expressed in inability to solve equations in this topic and to apply algebraic rules properly. However, a demand for a motivation which highlights the attributes of a topic or equation type is not propagated to lower levels.

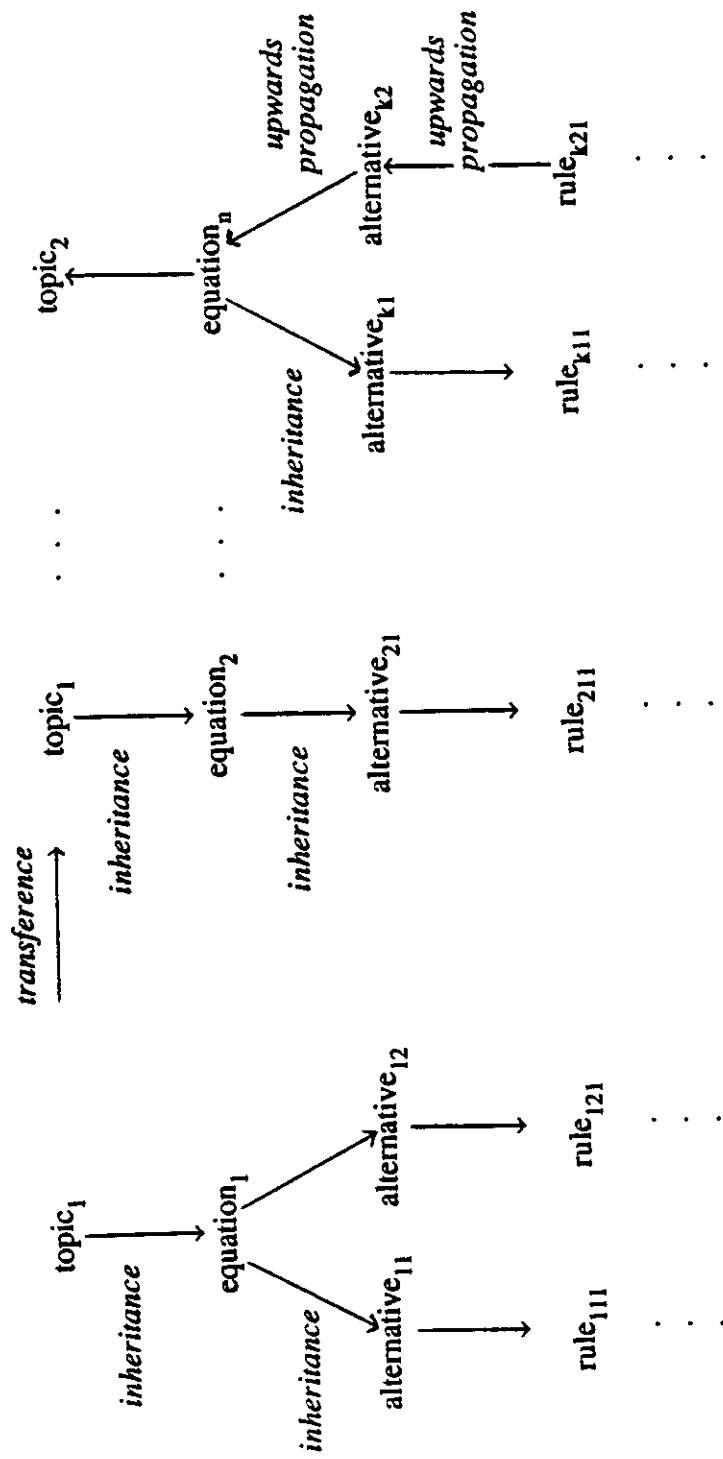


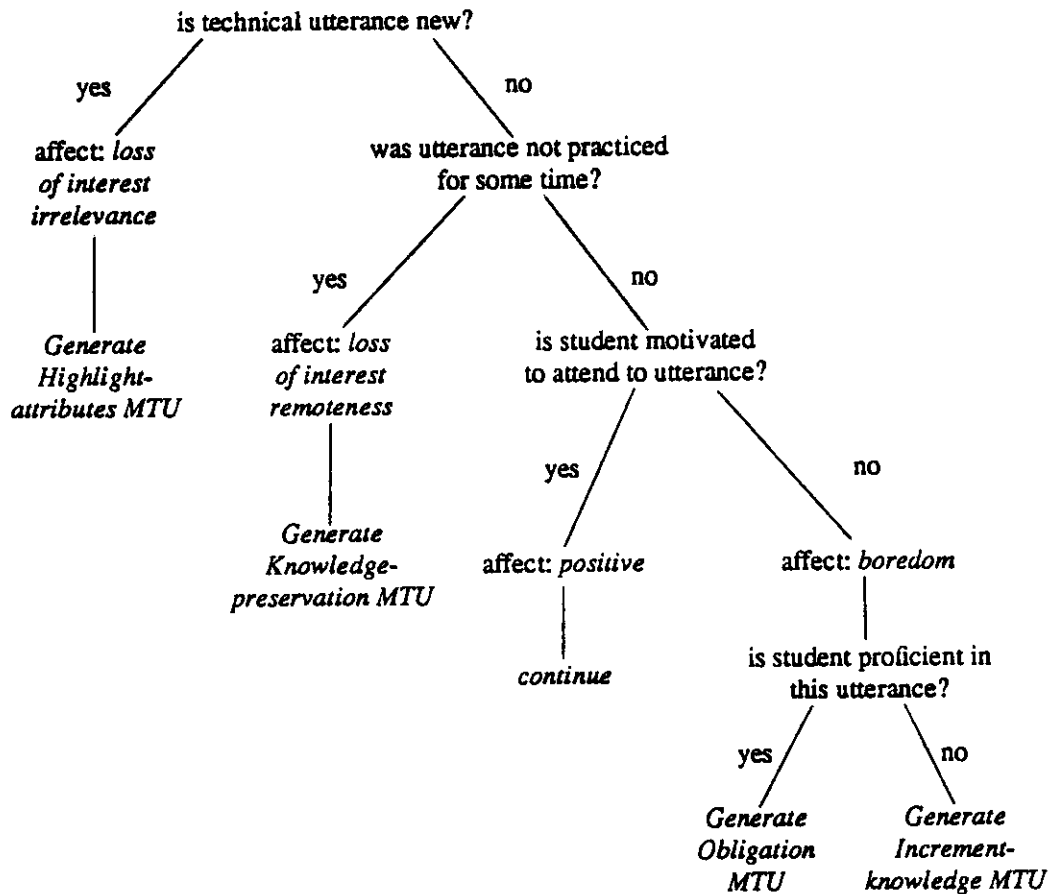
Fig. 4.6: Motivation Relations in a Context Hierarchy

Transference — If a student is motivated to attend to the solution of a given equation through the equation's topic, its pattern, or the techniques being used to solve it, then this motivation applies to the following equations which share the same characteristics.

Upwards propagation — A student may also be motivated to attend to an equation, by presenting it as a vehicle for the introduction of a technique, or by means of a knowledge-status related motivation for a previously discussed method. We shall denote this type of motivation *upwardly propagated*. If a new method is being studied, this motivation is propagated upwards to the equation, and may then be inherited by other alternatives, stating that they are presented "for comparison purposes," i.e., so that their performance may be compared to those of the alternative which includes the method in question. After all the alternatives have been presented, a comparison statement to this effect has to be issued, e.g., "The last alternative is faster and less error-prone than the two previous ones." If a knowledge-status related motivation is being used by the tutor, then if it is shared by all the solution alternatives, it may be propagated upwards. A motivational sentence like "This equation enables us to practice a couple of methods, which we have not seen for a while," is generated by propagating upwards the knowledge-preservation motivation of its alternatives.

In Fig. 4.7 we present the model used by the Comprehension-Processes Module to generate an appropriate motivational MTU.

According to this model, a student is presumed to experience loss of interest upon encountering a new technical utterance, until he is motivated by means of an MTU which highlights the attributes of this utterance. If the technical utterance has



**Fig. 4.7: Process for Generating a Motivational MTU**

already been discussed, the appropriate motivation depends on the student's knowledge status, i.e., if this utterance has not been studied for a while, a knowledge-preservation motivation is produced. If, on the other hand, the technical utterance has been practiced lately and the student was not recently motivated to attend to it, the system considers an increment-knowledge motivation. If this motivation is not applicable, the Comprehension-Processes Module uses the catch-all obligation (social) motivation.



Given a list of technical messages, FIGMENT applies the motivation build-up process to the topic and equation messages, and also to selected rule messages in each solution alternative. After the motivation requirements of each of these technical utterances have been ascertained, the system selects the motivation to be presented. Let us now consider the questions concerning a topic technical utterance shown in the discrimination net of Fig. 4.7.

*Was the technical utterance not practiced for some time?*

To answer this question FIGMENT takes into consideration both the student's last exposure to the technical utterance in question and the possible deterioration of his knowledge. The latter depends mainly on a student's talent.

IF { CURRENT-EQUATION-NUMBER — LAST-EXERCISE-EXPOSED-TO } >  
 DETERIORATION-FACTOR  
 THEN ASSERT *technical utterance was not practiced for some time*

Where:

DETERIORATION-FACTOR = max { MINIMUM-DETERIORATION-THRESHOLD,  
 TALENT × OUT-OF-PRACTICE-THRESHOLD }

MINIMUM-DETERIORATION-THRESHOLD is the minimum number of equations that must have elapsed before knowledge starts deteriorating.

OUT-OF-PRACTICE-THRESHOLD contains the number of equations that generally elapse until a student of talent 1 starts forgetting a particular topic.

*Is the student motivated to attend to the technical utterance?*

This question expresses the motivation-transference relation. If the student was recently motivated to attend to the current topic, then by the transference relation, this motivation remains valid; and by the inheritance relation, no motivation is required for the equation and the solution alternatives presented

up to its first successful one. The motivation to attend to a known technical utterance depends on a student's diligence and on his exposure to consecutive utterances of the same type. It is evaluated by the following rule:

IF { NUMBER-OF-EXERCISES-SINCE-MOTIVATION +  
PREDICTED-NUMBER-OF-EXERCISES } < MOTIVATION-FACTOR  
THEN ASSERT *student is motivated to attend to technical utterance*

Where:

MOTIVATION-FACTOR = max { MINIMUM-MOTIVATION-THRESHOLD,  
DILIGENCE × REQUIRED-MOTIVATION-THRESHOLD }

MINIMUM-MOTIVATION-THRESHOLD is the minimum number of equations that must have elapsed between two motivations for the same topic.

REQUIRED-MOTIVATION-THRESHOLD is the number of equations the most diligent student is willing to solve in a topic, without being motivated.

The product of DILIGENCE×REQUIRED-MOTIVATION-THRESHOLD expresses the number of equations a particular student is willing to solve without requiring motivation. This number has to be adjusted by means of the MINIMUM-MOTIVATION-THRESHOLD, in order to avoid motivating a very lazy student for each equation.

*Is the student proficient in the technical utterance?*

The proficiency of a student depends on his relative expertise. The following rule is used to answer this question:

IF RELATIVE-EXPERTISE ≥ KNOWLEDGE-THRESHOLD  
THEN ASSERT *student is proficient in technical utterance*

The student's RELATIVE-EXPERTISE is defined as a ratio of the equations mastered and the equations he was exposed to; and the KNOWLEDGE-

THRESHOLD is a percentage of solved equations beyond which a student is considered to know a topic.

Once the adequate motivation type for the topic has been ascertained, FIGMENT considers the possibility of motivating a student through the pattern of the current equation. To this end, the above described process as activated, with the following modifications.

A highlight-attributes motivation may be presented only for an equation with a distinguished and unqualified pattern (see Appendix 2).

If a requirement for a knowledge-status related or social motivation for the topic has been recorded, but the student was recently motivated to attend to an equation with the current pattern, then the student is considered to be motivated to attend to the current equation, and the requirement for a topic motivation is canceled.

If the equation is evaluated by the system as being of low level of difficulty for the student, then even if the student lacks expertise in solving this type of equation, an increment-knowledge motivation cannot be presented, and it has to be replaced by an obligation motivation.

An equation with a qualified pattern and an equation without a distinguished pattern may be motivated only if they inherited a need for motivation from the topic. In this case, only an obligation motivation should be generated, since the Problem Solving Expertise Module does not store any attributes for these patterns.

Finally, the Comprehension-Processes Module has to ascertain that the student is motivated to attend to each solution alternative. In order to accomplish this task, we assume that each alternative is characterized by a *typical sequence* of one or more methods, which set it apart from other alternatives. Usually a typical sequence is applied at the beginning of an alternative, and after it is completed, the rest of the steps are rather routine. A motivation has to be presented only for the method or methods in the typical sequence corresponding to each alternative. To generate such a motivational MTU, the following rule, which accounts for the inheritance relation, is incorporated in the process described in Fig. 4.7.

IF a rule is not new

AND the equation does not require a knowledge-status related  
or social motivation

AND the equation has not been solved yet

THEN the method is considered to be motivated

Unlike the process applied for motivating a student to attend to the solution of an equation, an inherited requirement for a knowledge-status related motivation is not canceled if a rule was previously motivated. This is because there may be more than one rule in a typical sequence, and in order to cancel a motivation requirement for an alternative, all the rules in its typical sequence have to be previously motivated.

#### 4.5.1 Selecting a Motivation

After the motivation build-up process has been activated on the topic, equation and the typical sequence of methods in each solution alternative, a structure contain-

ing the resulting motivation MTU-codes is produced. For example:

TOPIC	<i>knowledge-preservation</i>
EQUATION	<i>highlight-attributes</i>
ALTERNATIVE <sub>1</sub>	(method <sub>1</sub> <i>increment-knowledge</i> ) (method <sub>2</sub> <i>increment-knowledge</i> )
ALTERNATIVE <sub>2</sub>	(method <sub>1</sub> <i>obligate</i> )

In the process of selecting a motivation FIGMENT tries to imply the least possible lack of knowledge. Therefore, it will generally favour a motivation which highlights the attributes of a technical utterance over a knowledge-status related motivation. Among the latter, it will prefer an equation motivation over a topic motivation, and an upwardly-propagated method motivation over an equation motivation. In addition, FIGMENT prefers a knowledge-status related motivation to a social motivation. For the above-presented structure the system will highlight the attributes of the equation. These attributes are extracted from the Problem Solving Expert's domain knowledge, producing a sentence such as "Let us consider a new type of equation, which is very important and is encountered frequently."

Occasionally this policy will not be applied. For instance, if the current topic requires motivation and the number of equations is being advertised, a topic motivation is mandatory. This policy may also be violated for aesthetical reasons, i.e., to avoid repeating the same type of motivation. (For a detailed presentation of the motivation-generating rules, see Appendix 7.)

If both a topic and an equation require a highlight-attributes motivation, then both motivations may be presented. This yields an introduction like the following:

"Let us consider the important topic of quadratic equations. Here is an equation:

$$x^2 - 3x + 5 = 0$$

This type of equation is very common.”

If a highlight-attributes motivation of a method is upwardly propagated, then an equation may be introduced as follows: “This equation enables us to introduce the very important and useful technique of substitution, but first let us consider two other ways of solving this equation, for comparison purposes.” The adversative MTU “but” is produced because the Tutoring Strategist postpones the introduction of a new technique to the last alternative, thus violating the expectations established in the first part of the sentence.

If the same type of motivation appears at least once in all the alternatives, then an upwardly propagated knowledge-status related motivation may be generated. If both types of knowledge-status related motivation are applicable to all alternatives, the type which appears more times is selected. For example, given the following MTU-codes for typical rule sequences of three alternatives, the increment-knowledge motivation should be selected, yielding a sentence such as “Through this equation we are able to exercise a few techniques which demand some more practice.”

ALTERNATIVE<sub>1</sub> (method<sub>11</sub> *increment-knowledge*)  
 ALTERNATIVE<sub>2</sub> { (method<sub>21</sub> *increment-knowledge*),  
 (method<sub>22</sub> *knowledge-preservation*)  
 ALTERNATIVE<sub>3</sub> { (method<sub>31</sub> *knowledge-preservation*),  
 (method<sub>32</sub> *increment-knowledge*)

Finally, after the motivation MTU-code for an equation has been selected, each alternative which is not automatically motivated through the inheritance relation,

has to be separately motivated. The following sentences accomplish this task:

1. "Another alternative through which we can practice a couple of techniques we have not seen for a while consists of the following operations."

This motivation is issued if the equation was motivated through an upwardly propagated knowledge-status related motivation.

2. "Through the following alternative we shall introduce a very interesting technique, namely Patt's guessing method for solving quadratic equations."

This sentence highlights the attributes of a forthcoming method.

3. "Another solution I would like you to try consists of the following steps."

This sentence contains a social motivation, however it may also be used when a new method is being introduced. In this case, the attributes of this method are highlighted later on.

4. "The second alternative enables us to exercise a few rules which demand plenty of practice."

This sentence contains a knowledge-status related motivation.

The motivation-selection process presented in Fig. 4.7 is an integral part of the comprehension-processes activated by a student when attending to a technical utterance. Nevertheless, this process together with the motivation-selection rules should be activated by the Tutoring Strategist, in conjunction with the didactic rules which select the material to be presented. This will enable the Tutoring Strategist to present topics, equations and techniques which do not have to be motivated by means of a social motivation.

## 4.6 Partial Output of the Comprehension-Processes Module

This section describes the Knowledge-Acquisition and consolatory MTUs produced by the Comprehension-Processes Module. After the preceding processes have been activated, the Comprehension-Processes Module produces a list of extended messages. Each message contains a technical utterance accompanied by codes which specify requirements for MTUs. Figure 4.9 depicts the MTU requirement-codes generated from the output of the Tutoring Strategist presented in chapter 3 (repeated in Fig. 4.8 for convenience). These MTU-codes correspond to the second quadratic equation presented after a sequence of equations of a different type, to a student who is very competent but not diligent.

The first entry in Fig. 4.9 contains a topic technical utterance accompanied by a CONTINUE verification implementational MTU-code and an OPEN focal MTU-code. The former is produced by the Comprehension-Processes Module since the current equation is the second quadratic equation, i.e., the student is not familiar with this type of equation. The latter is generated to preclude confusion due to the presentation of a topic. FIGMENT does not produce a focus-close MTU, since there is no open focus.

The second entry requires a focus-open and a length-related MTU for the equation. The focal MTU is generated to introduce the forthcoming equation. The length-related MTU is required by a student who is not very diligent, since three alternatives for solving the equation are to be discussed. Notice that the code EXIST is produced, because these are all the possible alternatives.



TOPIC: quadratic  
EQUATION:  $(x-3)^2 - 4(x-3) - 12 = 0$

(ALTERNATIVE 1)

RULE: factor out x-3 from terms 1 and 2  
PATTERN: x-3 is a factor common to terms 1 and 2  
EXPECTATION: get a factor common with rest of equation  
RESULT:  $(x-3)(x-7) - 12 = 0$

RULE: remove parentheses  
RULE: collect terms  
RESULT:  $x^2 - 10x + 9 = 0$   
CONTINUE

(ALTERNATIVE 2)

RULE: remove parentheses  
RULE: collect terms  
RESULT:  $x^2 - 10x + 9 = 0$   
CONTINUE

(ALTERNATIVE 3)

RULE: substitute  $y = x-3$   
PATTERN:  $\left\{ \begin{array}{l} x \text{ appears only in expression } x-3 \\ \text{and } x-3 \text{ appears more than once} \end{array} \right.$   
PURPOSE: get canonic expression  
RESULT:  $y^2 - 4y - 12 = 0$

RULE: quadratic formula  
RESULT:  $y=6$  or  $y=-2$

RULE: substitute back x-3 for y  
RESULT:  $x-3 = 6$  or  $x-3 = -2$

RULE: transfer term  
RESULT:  $x=9$  or  $x=1$   
FINISH

Fig. 4.8: Sample Input to Comprehension-Processes Module

In the third entry, a focus-open MTU-code is generated for the first alternative, since the number of alternatives was mentioned previously, and in the absence of an introduction the student may temporarily mistake the first operation for the first alternative. The expectation corresponding to the factor-out rule is accompanied by an implementational MTU, since the student has seen this statement once before (entry No. 4). Next, the entry corresponding to the remove-parentheses rule contains three MTU-codes. This rule was encountered twice before, therefore the Comprehension-Processes Module generates an implementational MTU for it. Since the student is very talented, the system finds it necessary to issue a complexity-related MTU which signals the ease of this rule. The remove-parentheses rule is not inherently long, however, since it is being applied several times, it requires enough computations to warrant the generation of a length-related MTU for a student who is not diligent.

The second alternative (entry No. 6) requires a focus-close MTU, since the first alternative is still in focus, and a focus-open MTU, to introduce it. In addition, since the first alternative solved the equation, the second alternative has to be motivated. To this effect, the Comprehension-Processes Module selects an ATTEMPTED motivation for lack of a knowledge-status related motivation. The Sentence Composer generates a sentence such as "Another approach you may have considered consists of the following operations," from this MTU-code. As before, the remove-parentheses rule requires several computations, therefore, a situation-dependent length-related MTU has to be generated. Since the current usage of the remove-parentheses rule does not resemble its usage in the previous alternative a LIKE-BEFORE implementational MTU is not required, and since the implementational MTU generated in the previous alternative already put the student in verification-mode regarding his knowledge of this rule, a KNOWN MTU would be superfluous.

The third alternative also requires a focus-close and focus-open MTU, and a motivation. This time, however, a HIGHLIGHT-ATTRIBUTES motivation is presented, since the expression-substitution rule is being introduced. Finally, a complexity-related MTU advertising the ease of this rule is generated.

Utterance	MTU type	MTU Code
TOPIC	{ <i>Focus?</i> <i>Implementation Mode?</i>	(OPEN) (CONTINUE)
EQUATION	{ <i>Focus?</i> <i>Length?</i>	(OPEN) (EXIST 3)
ALTERNATIVE <sub>1</sub>	<i>Focus?</i>	(OPEN)
EXPECTATION	<i>Implementation Mode?</i>	(KNOWN 1)
RULE (Remove Parentheses)	{ <i>Implementation Mode?</i> <i>Complexity?</i> <i>Length?</i>	(KNOWN 2) (EASY) (LONG-SITUATION)
ALTERNATIVE <sub>2</sub>	{ <i>Focus?</i> <i>Motivation?</i>	(CLOSE 1) (OPEN) (ATTEMPTED)
RULE (Remove Parentheses)	<i>Length?</i>	(LONG-SITUATION)
ALTERNATIVE <sub>3</sub>	{ <i>Focus?</i> <i>Motivation?</i>	(CLOSE 2)(OPEN) (HIGHLIGHT-ATTRIBUTES) (Substitute Expression)
RULE (Substitute Expression)	<i>Complexity?</i>	(EASY)

**Fig. 4.9: Requirement-codes for Knowledge-Acquisition MTUs for Sample Input**

These codes for Knowledge-Acquisition MTUs are complemented by means of codes for Knowledge-Organization and affect-transference MTUs, whose generation is described in the following chapter. The English-representation produced by the Sentence-Composer from this output is also presented in the next chapter.

## CHAPTER 5

### Generation of Knowledge-Organization MTUs

In this chapter we describe the generation of Knowledge-Organization and affect-transference MTUs. As stated above, they are derived from information in the domain knowledge of the Problem Solving Expert and from the structure of the Tutoring Strategist's output.

#### 5.1 Generation of Additive MTUs

The additive MTUs generated by FIGMENT represent direct translations of the structure of the output of the Tutoring Strategist. Therefore, it will suffice to present some examples which illustrate common usages of these additive MTUs in discourse related to the solution of algebraic equations:

1. "Next, we remove parentheses, collect terms **and** apply the quadratic formula."

The *simple additive* MTU "and" represents a direct translation of a sequence of rules.

2. "You can generally solve quadratic equations by removing parentheses, collecting terms, **and then** applying the quadratic formula **or** completing the square."

The simple additive MTUs "or" and "and then" are produced by translating the information in the general-method slot corresponding to the topic of

quadratic equations (see section 3.1.1). This information has the following structure:

*((remove-parentheses collect-terms (quadratic-formula complete-square)))*

In this representation, alternative approaches appear between odd-numbered parentheses, and sequences of rules, between even-numbered parentheses. Thus, there is only one general approach for solving quadratic equations, which consists of three rules applied consecutively, and there are two alternatives for the third rule.

3. “This topic is very interesting **and** important.”

The MTU “and” directly reflects the relationship between the attributes selected by the Comprehension-Processes Module for this motivational MTU (see section 4.5). Since both attributes are positive, an additive MTU is issued; whereas if one of the attributes were negative, an adversative connective such as “but” would be required. For a detailed explanation on the choice of connective see section 6.1.

4. “As stated above, this technique is very useful. **Furthermore**, it is elegant **and** efficient.”

This example is similar to the previous one, but for the fact that it features the *internal additive* MTU “furthermore,” in addition to the external MTU “and.” As before, the attributes are represented as a list, and the choice of MTUs depends on their relationships. In this example, the attribute-list is divided into two sentences, because the attribute “useful” is modified by the *implementational* MTU “as stated above,” while the other attributes are not.

*Realization of expectation* MTUs also belong to the additive category. However, unlike the MTUs presented above, their generation is not straightforward, as they depend on the expectations triggered in a student by previous discourse. These MTUs are generated by a module which monitors the status of the student's expectations, and determines whether a realization or a violation of these expectations has occurred.

## 5.2 Generation of Adversative MTUs

In this section we discuss the generation of *realization* and *expectation violation* MTUs (e.g., "however," "nevertheless," "despite this," etc). Affect-transference MTUs which are frequently linked to the violation of expectations, shall be produced by the same algorithms which produce expectation-violation MTUs. First, we shall consider some of the expectations experienced by a student. Thereafter, we shall examine the impact of different types of technical utterances on these expectations. The following types of expectations are recognized by FIGMENT:

i. *Expectation of the existence of a solution.*

The presentation of an equation triggers an implicit expectation of the existence of a solution. This type of expectation is activated in a more explicit manner by the introduction of a solution alternative or the presentation of a rule.

ii. *Expectation of a particular result.*

This type of expectation is usually explicitly prompted by a teacher, by means of an expectation technical utterance. For example, "We factor out  $(2x+5)$  from the first two terms, **hoping** to get a factor common to the rest of the

equation.” At each point in the solution of an equation there is only one active expectation of this type. The system records an expectation for hopes only, since a realization-of-expectation MTU following the statement of a purpose would be superfluous.

iii. *Expectation of similarity with previous solution paths.*

A tutor triggers this type of expectation by presenting a student with an implementational MTU which states that the current equation or result is similar to a previous equation. For instance: “This equation reminds us of equations 1 and 3.” This type of statement prompts a student to expect to solve the current equation, by repeating the algebraic operations which were successful previously. We distinguish between two types of similarity expectations, according to their scope:

**Inter-alternative** — If the similarity statement refers to an equation, a student would expect the solution alternatives considered for the current equation to have been applied earlier; and

**Intra-alternative** — If the similarity statement concerns an intermediate result, arrived at during the solution of an equation, the student expects the subsequent operations in the current alternative to be equal to the ones used to solve the equation being recalled. This expectation is not triggered if an inter-alternative expectation is already active, and does not transcend to any other approaches applied to solve the current equation.



Another type of expectation found in this category is the *expectation of dissimilarity*. This expectation is triggered by a qualified implementational MTU (see Appendix 2). After mentioning the similarity between the current equation and a previous one, this MTU points out a discrepancy between them. For example, "This equation is similar to the last one, **but it contains an additional term.**" This type of statement would cause a student to expect the solution paths of the equation under consideration to differ from the solution paths for the previous equation.

iv. *Expectation of a particular technical utterance.*

If a teacher mentions that he will discuss a particular technical utterance, a student would expect this utterance to be immediately issued. For example, this type of expectation is activated by means of the following statement: "The following alternative serves to introduce the method of ... ." It differs from the previous expectations, in that when an utterance is expected, its arrival is not in question, but only the time of its arrival, whereas when a mathematical event is expected, its fulfillment is uncertain.

Let us now consider the effect of each technical utterance and its accompanying MTUs on these expectations.

i. *Equation.*

As stated before, an equation technical utterance by itself triggers only an implicit expectation of the existence of a solution. However, if it is accompanied by an implementational MTU such as "This equation is similar to the previous one," the student will expect to see the solution alternatives which succeeded in solving the previous equation (i.e., expectation of similarity). In

addition, if a length-related MTU like the following is issued: "There are three ways to solve this equation," an explicit expectation of the existence of solution would be triggered in the student.

ii. **Alternative.**

The introduction of an alternative by means of a focal MTU triggers an expectation of the existence of a solution. If an expectation of an absence of solution was active, the introduction of the alternative represents a violation of the student's expectations, and entails the generation of an expectation-violation MTU. In addition, if the tutor can foresee that this alternative will solve the equation, the generation of a positive affect-transference MTU is in order, yielding a sentence like the following: "Fortunately, however, there is a way to solve this equation." To avoid generating boring and repetitious text, this affect-transference MTU can be generated only if a negative affect-transference MTU was not produced for the utterance which caused the expectation of the absence of a solution.

iii. **Rule.**

Like for alternatives, if a previous technical or meta-technical utterance caused a student to expect the current equation to have no solution, the mere presentation of a rule violates this expectation. Thus, like before, an expectation-violation MTU has to be generated in this situation, yielding a sentence like the following: "Nevertheless, we can still remove parentheses."

If there is an active expectation of similarity, triggered by an implementational MTU, the following rules govern the generation of realization or violation of expectation MTUs.

If the rule applied in the current equation is indeed equal to the one used to solve the equation referenced in the implementational MTU, and an MTU signaling the realization of a similarity expectation was not previously produced, then it has to be generated. The rationale supporting this rule is that once an expectation of a particular solution path is realized by the first step in the solution of an equation, the tutor does not have to continue reinforcing this realization of expectations. According to this policy, the implementational MTU "This equation is similar to the previous one" might be followed by an MTU like "Indeed, we shall begin by applying the same solution method."

If the rule applied in the current equation was not used in the equation referred to by the implementational MTU, then a similarity expectation violation MTU has to be generated, yielding the following text: "However, we shall solve it by ... ."

An expectation of dissimilar solutions is activated by means of a qualified implementational MTU such as "This equation reminds us of equation #3, but it has an additional term." This expectation is violated, if the first algebraic operation applied to solve the equation resembles the first operation used to solve the referenced equation. In this case, a statement like "Still, we begin by ... ," replaces the expectation of dissimilarity with a similarity expectation. If the expectation of dissimilarity is realized, it is canceled.

As shown in Appendix 2, a rule technical message may contain a list of equations in which this rule was used under similar circumstances. FIGMENT ascertains the status of an expectation of similarity or dissimilarity, by testing

whether the current list is a superset of the list of equations which triggered this expectation. As long as an expectation of similarity remains active, no implementational MTUs shall be produced for a rule, since they would only repeat the information transmitted in a previous implementational MTU.

Finally, a rule also affects an expectation of a particular technical utterance, since the rule following an introduction may not be the expected one. In this case, an event violation MTU like the following is generated: **“But, before we apply this method, we remove parentheses.”**

iv. **Result.**

If there is an active expectation of a particular result, then the result of each algebraic operation has to be checked against this expectation. If the result violates or realizes the expectation, an MTU to this effect has to be generated, and the expectation is canceled. The following sentence illustrates a realization of expectation MTU: **“This method arrives at the result we were hoping for.”** If the violation of an expected result disables the solution of the equation, then a negative affect-related MTU should be generated, yielding a sentence such as **“Unfortunately, however, we get ... .”**

An algebraic operation may constitute an intermediate step, which neither violates nor realizes an expectation. Thus, no expectation-related MTUs are issued for it, and the expectation remains active. If the relevant result is reached some time after the expectation was stated, then the expectation might have to be restated, yielding the following sentence: **“We were hoping to get a common factor between some of the terms in the equation, however, we arrived at ... .”**

A result technical utterance may also affect the status of an expectation of similarity. However, it can cause only a violation of this expectation, since such an expectation would be realized by a preceding rule. Let us illustrate this point by means of the following example:

“This equation is similar to equations 1 and 3. **Indeed**, we start solving it by factoring out common factors. **However**, we get ... ”

In the first sentence, an expectation of similarity is triggered. It is realized in the second sentence, containing a rule technical utterance. The third sentence constitutes a violation of this expectation, and the expectation of similarity is canceled. If the violation of an expectation of similarity occurs after several steps have been performed, a more explicit expectation violation MTU may be required, for example: “**Unlike the result obtained in equations 5 and 6**, we get ... .”

As stated before, an expectation of dissimilarity exists only at the beginning of a solution. Therefore, by the time a result is presented, this expectation is no longer active.

v. Method.

These technical utterances differ from the above mentioned utterances in that they may refer to a topic or equation type which are not currently in focus. Therefore, any expectation triggered by a digressing method technical utterance is a *temporary expectation*, which has no impact on the expectations related to the items in the permanent context, and is canceled when the discourse returns to the main subject. Similarly, the expectations regarding the

items in the permanent context have no effect on a temporary expectation. The following rules govern the generation of expectation and affect-transference MTUs for method technical utterances.

If a solution is being expected for a certain topic or equation type, and the method technical utterance contains information regarding the lack of a general or a specific method, then the expectation of the existence of a solution is canceled. If the expectation of the existence of a solution was explicitly activated, then an expectation-violation MTU is in order. In addition, the system determines the need for an affect-transference MTU by applying the following rule:

If a technical utterance concerns a general solution method, and the expectation of a solution was implicitly established or the technical utterance which explicitly established this expectation was not accompanied by an affect-transference MTU, then a negative affect-transference MTU may be issued.

This rule is based on the belief that if there exists a general method for solving an equation, the absence of specific methods has no affective impact on a student. However, in the absence of a general technique, this rule may produce the following text: "Unfortunately, we shall not examine a general technique for solving third degree equations."

If a solution is not being expected, and there exists a solution method, then an expectation-violation MTU is generated, and the expectation of the

existence of a solution is activated. In addition, if the method technical utterance contains information regarding particular techniques to solve an equation and the technical utterance which canceled the expectation of the existence of a solution was not accompanied by an affect-transference MTU, then a positive affect-transference MTU should be produced.

The following sentence is generated by means of this rule: "Fortunately, however, we can solve certain types of third degree equations by factoring out common factors, or alternatively, applying the appropriate factorization formula." The system does not produce positive affect-transference MTUs for general methods, since their existence is not considered a fortuitous event.

vi. Finish.

A finish technical utterance signals the termination of a solution path. It cancels any active expectation of a particular result and also any intra-alternative similarity expectation. If the current alternative did not succeed in solving the equation, and an expectation of the existence of a solution is still active, the expectation is canceled and a negative affect-transference MTU is generated, yielding a sentence such as "Unfortunately, there is nothing else we can do here." If, however, a solution is no longer expected, due to a previous failure of an expectation of a particular result, no affect-transference MTUs need to be produced.

In addition to these technical utterances, there are some configurations of Knowledge-Acquisition MTUs, which entail the generation of expectation-violation

connectives, in particular the connective “but.” The generation of these MTUs and their connectives is performed by the Sentence Composer. Since this process is explained in detail in Chapter 6, it will suffice to present here a few examples:

1. “I know these are many alternatives, **but** you shall benefit from learning the methods they illustrate.”

In this example, the second sentence contains a consolatory MTU, which counteracts the misgivings experienced by a student due to the presence of many alternatives, advertised by a length-related MTU.

2. “There is another way of solving this equation, **but** it entails many steps.”

In this example, the first sentence represents a focal and a temporal MTU, and the second sentence, a length-related MTU. Whereas the former reaffirms the expectation of the existence of a solution, the latter imperils the realization of this expectation.

3. “This method is very important **but** simple.”

An expectation-violation MTU is issued in this example, since, in most cases, a reader would not expect to encounter these attributes in reference to the same object.

4. “This equation enables us to introduce a useful technique, **but first**, let us consider three other approaches, for comparison purposes.”

In this example, the second sentence temporarily negates the occurrence of the event advertised by the first motivational sentence. Therefore, an MTU which violates an event expectation has to connect between them.



Expectation-violation MTUs may also be applied to estimational, implementational and motivational MTUs, in relation to expectations triggered by previous technical utterances. For example, “Unlike the previous equation, this equation is quite simple.” However, due to the partial short-term memory supported by the present version of FIGMENT, this configuration of MTUs is not generated.

Dismissal MTUs (e.g., “in any case,” “either way”) are used by a human tutor when two different but rather short paths in the solution of an equation arrive at the same pattern. The following text illustrates this usage: “We may transfer terms first and then collect terms, or we may first collect terms on both sides of the equation and transfer terms thereafter. **Either way**, we arrive at ... .” Since the current implementation of FIGMENT assumes that the output of the Tutoring Strategist has a tree structure, it does not generate this type of MTU.

A recognition MTU is issued when an implicit pattern is recognized and made explicit. For example: “**Notice that** we can rewrite this expression as ... .” The generation of this type of MTU is linked to the application of rewrite rules. These rules provide a different representation for a given expression by means of parentheses (e.g., “ $-x+1 = -(x-1)$ ,” “ $x^2-2x+1 = (x-1)^2$ ”).

### 5.3 Generation of Causal MTUs

As stated in section 2.3.1, most of the relations expressed by Causal MTUs are invariant with respect to equations and are inherent in the algebraic rules. Therefore, they can be directly represented by the following slots of the Mathematical and Planning Knowledge component in each rule (see section 3.1.1).

i. Pattern and Preconditions.

These slots contain the reason or justification for applying a particular rule.

The following example illustrates the generation of a reason MTU:

Given the equation:

$$x(x-5) - 4(x-5)^2 = 0$$

And the rule instantiation:

RULE: factor out x-5 from terms 1 and 2  
PATTERN: x-5 is a factor common to terms 1 and 2

FIGMENT may generate any of the following sentences:

1. "Since x-5 is a factor common to both terms, we factor it out."
2. "We factor out x-5 because it is a factor common to both terms."
3. "x-5 is a factor common to both terms. Therefore we factor it out."

ii. Purpose/Expectation.

If a particular result of an algebraic operation is expected with certainty, the purpose of this operation is stated. If, however, the result is uncertain, then it is hoped for. We shall illustrate the generation of purpose and hope MTUs by means of the following examples.

Given the previous equation, and the following rule instantiation:

RULE: factor out x-5 from terms 1 and 2  
EXPECTATION: get a product of factors (1)\*  
\*(degree of certainty)

FIGMENT may generate the following text: "We factor out x-5 from both terms, in order to get a product of factors."

The following equation is quite similar to the previous one, but contains two additional terms:

$$x(x-5) - 4(x-5)^2 + 3x - 20 = 0$$

The expectation instantiated in this case would be:

EXPECTATION: get a factor common with the rest of the terms (0.5)

And the following text would be produced: "We factor out  $x-5$  from the first and second terms, **hoping to get a factor common with the rest of the terms.**"

If both, the pattern and the expectation of a rule are to be stated, then the generated text incorporates both types of MTUs:

1. "Since  $x-5$  is a factor common to the first and second terms, we factor it out, **hoping to get a factor common with the rest of the terms.**"
2. " $x-5$  is a factor common to the first and second terms, **therefore we factor it out. We perform this operation hoping to get a factor common to the rest of the terms.**"
3. "We factor out  $x-5$  from the first and second terms, **hoping to get a factor common to the rest of the terms. We are able to perform this operation because  $x-5$  is a factor common to the first and second terms.**"

iii. Method Description.

The slot which contains a detailed description of the application of an algebraic rule requires the generation of a means MTU such as "This can be

accomplished in the following way.”

iv. **Correctness.**

A correctness MTU like “this works because,” signals the presentation of the explanation or correctness proof for an algebraic rule. The information in this slot, and its corresponding MTU are not generated by FIGMENT at present.

The result of an algebraic operation is extracted from the output of the Problem Solving Expert. If it constitutes a realization of an expectation of a particular result, the following sentence is produced: “We factor out  $x-5$  from the first and second terms, yielding the result we were hoping for.” If there is no previous expectation, a sentence like the following is generated: “This operation arrives at the following result.”

#### **5.4 Generation of Attributive MTUs**

In the current implementation, attributive MTUs are generated for the methods used to solve equations belonging to a given topic or having a distinguished pattern. A group of methods is either generally applicable, or may be applied only under certain circumstances. Like most of the technical utterances discussed in the previous section, a method technical utterance is extracted from a slot in the domain knowledge. This slot directly represents the appropriate attributive MTU. The following examples illustrate the usage of these MTUs:

“**In general**, we can solve quadratic equations by removing parentheses, collecting terms, and then applying the quadratic formula or completing the square.”

“We can solve **certain types of** third degree equations by factoring out common fac-

tors, or alternatively, applying the appropriate factorization formula.”

## 5.5 Generation of Temporal MTUs

FIGMENT produces temporal MTUs for equation, alternative and rule technical utterances.

### i. Equation.

If the number of equations to be solved was advertised, the first few equations should be accompanied by a temporal MTU. For example, “We shall discuss several linear equations, let us **begin with the following one.**” After a certain number of the advertised equations has been presented, a temporal MTU is no longer required and subsequent equations may be introduced by means of a focal MTU only. If the number of equations is not mentioned, temporal MTUs should not be generated, since they do not fit naturally in FIGMENT’s discourse. In this case, a sentence like “We shall now examine the **third** equation,” is meaningless.

### ii. Alternative.

A sentence generated to introduce a solution alternative usually contains a temporal MTU. For instance, “Let us consider **another** approach for solving this equation” or “**The second** alternative consists of the following steps.” These MTUs advertise the order in which the different alternatives were attempted. Before discussing their generation, we would like to distinguish between two types of temporal MTUs, according to the circumstances in which they are generated.

Ordinal — If the number of solution alternatives to be discussed was advertised in advance by a length-related MTU of the equation, then a tutor would generally use an *ordinal* number when introducing an alternative. For example: “The **second** approach to solving this equation consists of the following steps’”; and

Cardinal — If, on the other hand, the number of solution alternatives was not previously discussed, a temporal MTU like the following should be issued: “another approach,” “a different way to solve this equation,” etc.

The first solution alternative is usually treated differently than the rest. For instance, a teacher might begin discussing it by directly presenting the first algebraic operation performed. Therefore, in order to emulate human behaviour, the policies implemented for the first approach differ from the ones applied for the rest. The following rules generate a temporal MTU-code for the first alternative.

If the number of solution alternatives to be discussed was previously advertised by means of a length-related MTU for the equation, then, if more than one solution alternative are to be presented, an ordinal temporal MTU has to be generated. Otherwise, if only one approach is to be examined, a temporal MTU will be omitted, since its generation would only repeat the previous MTU. The first part of this rule yields a sentence like the following: “Let us consider the **first** approach to solve this equation.”

If the number of alternatives was not advertised in advance and a focal MTU was generated for this alternative, then, if more than one alternative shall be presented, a cardinal temporal MTU may be issued, yielding a sentence such as "One way to solve this equation consists of the following operations."

The generation of temporal MTUs for the rest of the alternatives is governed by the following rules:

If the current alternative is the last to be discussed, and there are more than two alternatives, then an MTU indicating this fact should be generated. For example, "Let us now consider the last alternative." This rule precludes the generation of the MTU "last" when only two alternatives are being discussed.

Otherwise, a temporal MTU is generated, according to the requirements for the generation of cardinal and ordinal MTUs.

iii. Rule.

The temporal MTUs generated for rule technical utterances differ from the ones produced for alternative technical utterances, e.g., "we begin by," "next," "afterwards," etc. Nevertheless, the policies which control their generation resemble the ones presented above.

If at least two algebraic rules have been presented, and the current rule yields the solution of the equation, then an MTU to this effect is generated. For instance: "we complete the solution by," "finally," etc. Notice that this policy refers to the last rule in the solution of the equation, as

opposed to the last rule being discussed.

If the entire solution alternative consists of one algebraic rule only, no temporal MTU has to be generated, since a temporal MTU would be redundant in this case.

## 5.6 Output of the Comprehension-Processes Module

The output of the Comprehension-Processes Module consists of a list of extended messages, which may contain requirement-codes for Knowledge-Organization, Knowledge-Acquisition and Affect-Maintenance MTUs (see Appendix 5).

Fig. 5.2 contains the completed MTU requirement-codes generated from the output of the Tutoring Strategist presented in Fig. 3.6 (repeated in Fig. 5.1 for convenience). The starred MTU-codes correspond to Knowledge-Organization MTUs, while the rest of the MTU-codes are used to produce Knowledge-Acquisition MTUs.

All the alternatives require a temporal MTU to signal the order in which they were attempted. Since the number of alternatives to be discussed was advertised, an ordinal temporal MTU is generated for the first and second alternatives. The third alternative is accompanied by the MTU "last." Likewise, temporal MTUs are generated for all the rules. Notice that in the first and second alternatives, the temporal MTU accompanying the last rule is its number, whereas the last rule of the third alternative has the MTU "last." This is due to the fact that the first two alternatives leave the last steps of the solution to the student, whereas the third alternative reaches the final solution. As stated above, causal MTUs are directly derived from the slots of the Mathematical and Planning Knowledge Component of each rule. Finally, the result



technical utterance for the factoring-out rule in the first alternative (entry No. 6) requires an adversative MTU which signals violation of the expectation of finding a factor common to the remaining term. This utterance establishes a transient expectation for the absence of a solution, which is in turn violated by the removing-parentheses rule in entry No. 7. Therefore, the removing-parentheses rule has to be accompanied by an expectation-violation MTU.

The Sentence Composer translates the technical messages and MTUs into the following text:

**We shall go on with the topic of quadratic equations. An equation follows:**

$$(x - 3)^2 - 4(x - 3) - 12 = 0$$

**There are three ways of solving this equation. The first alternative consists of the following operations:**

**First, since  $x-3$  is a factor common to the first and second terms, we factor it out from these terms. As you know, we perform this step hoping to get a factor common to the remaining term. However, it arrives at:**

$$(x - 3)(x - 7) - 12 = 0$$

**Nevertheless, we can still eliminate parentheses. We saw this method in equation number 4, it is easy, but requires a lot of computations in this case. Thereafter, we collect terms, yielding the following result:**

TOPIC: quadratic  
EQUATION:  $(x-3)^2 - 4(x-3) - 12 = 0$

(ALTERNATIVE 1)

RULE: factor out x-3 from terms 1 and 2  
PATTERN: x-3 is a factor common to terms 1 and 2  
EXPECTATION: get a factor common with rest of equation  
RESULT:  $(x-3)(x-7) - 12 = 0$

RULE: remove parentheses  
RULE: collect terms  
RESULT:  $x^2 - 10x + 9 = 0$   
CONTINUE

(ALTERNATIVE 2)

RULE: remove parentheses  
RULE: collect terms  
RESULT:  $x^2 - 10x + 9 = 0$   
CONTINUE

(ALTERNATIVE 3)

RULE: substitute  $y = x-3$   
PATTERN:  $\left\{ \begin{array}{l} x \text{ appears only in expression } x-3 \\ \text{and } x-3 \text{ appears more than once} \end{array} \right.$   
PURPOSE: get canonic expression  
RESULT:  $y^2 - 4y - 12 = 0$

RULE: quadratic formula  
RESULT:  $y=6$  or  $y=-2$

RULE: substitute back x-3 for y  
RESULT:  $x-3 = 6$  or  $x-3 = -2$

RULE: transfer term  
RESULT:  $x=9$  or  $x=1$   
FINISH

Fig. 5.1: Sample Input to Comprehension-Processes Module

$$x^2 - 10x + 9 = 0$$

**From here you can complete the solution of the equation by yourself.**

**Let us now examine the second way to solve this exercise. Through this alternative we can go over a solution, which you might have considered.**

**First, we get rid of parentheses. This technique demands plenty of calculations in this situation. Next, we collect terms, arriving at:**

$$x^2 - 10x + 9 = 0$$

**From this point you can obtain the solution by yourself.**

**We shall now consider the last alternative. This approach enables us to introduce the method of substitution, which is quite efficient.**

**First, we substitute  $y$  for  $x-3$ , because  $x$  appears only in expression  $x-3$  and  $x-3$  appears more than once in the equation. This technique is rather simple. Through it we get the following result:**

$$y^2 - 4y - 12 = 0$$

**We go on by applying the quadratic formula, arriving at:**

$$y = 6 \text{ or } y = -2$$

**We continue by substituting back  $x-3$ , yielding the following result:**

$$x - 3 = 6 \text{ or } x - 3 = -2$$

**Finally, we transfer the constant, namely  $-3$ , to the left hand side of the equation, obtaining:**

$x = 9$  or  $x = 1$

In the following chapter, we shall examine in detail the manner in which a text like the one presented above is produced by the Sentence Composer from the output of the Comprehension-Processes Module.

Utterance	MTU Type	MTU Code
TOPIC	{ Focus? Implementation Mode?	(OPEN) (CONTINUE)
EQUATION	{ Focus? Length?	(OPEN) (EXIST 3)
ALTERNATIVE <sub>1</sub>	{ Focus? *Sequence?	(OPEN) (ORDINAL 1)
RULE (Factor out x-3)	*Sequence?	1
EXPECTATION	Implementation Mode?	(KNOWN 1)
RESULT	*Result Expectation?	(VIOLATION)
RULE (Remove Parentheses)	{ Implementation Mode? Complexity? Length? *Solution Expectation? *Sequence?	(KNOWN 2) (EASY) (LONG-SITUATION) (VIOLATION) (2)
RULE (Collect Terms)	*Sequence?	(3)
ALTERNATIVE <sub>2</sub>	{ Focus? Motivation? *Sequence?	(CLOSE 1) (OPEN) (ATTEMPTED) (ORDINAL 2)
RULE (Remove Parentheses)	{ Length? *Sequence?	(LONG-SITUATION) (1)
RULE (Collect Terms)	*Sequence?	(2)
ALTERNATIVE <sub>3</sub>	{ Focus? Motivation? *Sequence?	(CLOSE 2) (OPEN) (HIGHLIGHT-ATTRIBUTES) (Substitute Expression) (LAST)
RULE (Substitute Expression)	{ Complexity? *Sequence?	(EASY) (1)
RULE (Quadratic Formula)	*Sequence?	(2)
RULE (Substitute Back)	*Sequence?	(3)
RULE (Transfer Terms)	*Sequence?	(LAST)

Fig. 5.2: Completed MTU Requirement-codes for Sample Input

## CHAPTER 6

### Generation of an English-Language Representation: The Sentence Composer

The Sentence Composer organizes the output produced by the Comprehension-Processes Module into paragraphs and sentences, and converts it into English. It has been mainly designed as a tool to test the feasibility of the ideas expounded in Chapters 4 and 5 and is composed of three major components: a Phrasal Dictionary, an Attribute-clause Generator and Utterance Generators.

#### Phrasal Dictionary.

This component generates expressions and words commonly used in technical discourse, in particular in the area of algebra. In addition, it performs the derivation and generation of verbs and nouns, and generates articles and pronouns. The generation of verbs, nouns and articles follows the rules of English Grammar, and need not be discussed. The generation of pronouns, however, is performed according to some rather simple but effective heuristics, and shall be explained in section 6.1.3.

#### Attribute-Clause Generator.

Utterances like “this method is very useful and interesting” or “we have never encountered this topic before,” frequently appear in technical discourse. These utterances contain attributes of mathematical entities. The Attribute-clause Generator applies rhetorical rules to generate one or more attribute clauses for a given technical utterance.

## Utterance Generators.

The manner in which MTUs interact and the actual text generated for them, depend on the type of the technical utterance under consideration. This component contains a hierarchy of dedicated procedures, which apply rhetorical rules tailored for each type of technical utterance, to determine the order of presentation of technical and meta-technical utterances, decide on the manner in which the items under consideration shall be referenced, and perform the actual generation of text corresponding to an extended message.

### 6.1 The Phrasal Dictionary

At present, the phrasal dictionary consists of 242 entries. Each entry has the following format:

*(keyword type motivation-relation last-used word-form clause-form)*

A particular word is found by its *keyword*. Some entries activate a special process to generate their English representation, e.g., nouns have to be derivated according to their grammatical number, and verbs, by their grammatical person and tense. The *type* of an entry signals which process is to be activated. A null type indicates that a representation of the entry can be chosen without further ado.

A sentence such as “This method is important and simple” is grammatically correct, however, most people would communicate the same facts by means of the following sentence: “This method is important **but** simple.” We account for the difference between these sentences by postulating that a particular value of an attribute triggers expectations regarding the values of other attributes of the same entity. If this expectations are violated, then an adversative MTU has to be generated.

In technical discourse, the values of attributes of topics, equations, methods, etc, not only exhibit this property, but also reflect the ability of a tutor to motivate a student by means of these attributes. For instance, a student can be motivated to practice a particular method by stating that it is important and difficult. However, no tutor would coax a student to practice a method repeatedly by advertising its ease.

In a dictionary entry, the *motivation-relation* slot is used to express these relationships. We distinguish between three types of motivation relations.

Type 1 — This type of motivation relation can be used by a tutor in coaxing a student to additional practice and study. Attributes like “difficult,” “important” and “interesting” belong to this category.

Type 2 — This motivation relation represents the opposite of the previous one. It contains attributes like “easy,” “short” and “simple.”

Null — Attributes belonging to this category can still be used by a tutor for motivation purposes, however they are neutral with respect to the listener’s expectations. Attributes like “new” and “common” are of this type. Thus both of the following statements would be acceptable: “This topic is important and appears quite frequently in tests” and “This method is simple and is frequently used.”

In general, when speaking or writing, people are reluctant to repeat recently mentioned expressions. In fact, they will use synonyms or near synonyms rather than repeat themselves. The *last used* slot in a dictionary entry supports this property. It contains the last English representation generated for the dictionary entry under consideration, and the clause number in which it was presented. By means of this slot, the



system verifies whether a selected English representation was recently used, and if so, chooses an alternative representation. The representation stored in this slot differs from the generated one, in that the verbs are stored in their infinitive form, and the nouns, in their single form. This precludes the repetition of the same verb or noun in different forms.

Many concepts in the English language can be expressed both, by a few words, or by clauses or entire sentences. Since different situations may call for different types of representations, the dictionary distinguishes between *word form* and *clause form*. For example, the dictionary entry corresponding to “nevertheless” has the following values:

Word form: {*nevertheless, however, despite this*}  
Clause form: {{{*contrary*} to (*our-expectations*)}}

The words enclosed in parentheses are keywords whose English representation has to be generated separately. In addition, the values of some slots in the clause form representation may contain parameters which are replaced by input given by other components of the Sentence Composer. For instance, the dictionary entry corresponding to “new” has the following values:

Word form: {*new*}  
Clause form: {(\$*person* have (*never*) (*study 1p pp*) \$*subject* (*before*))}

In this example, the parameters *\$person* and *\$subject* are to be supplied by the calling component. If they are absent, then the corresponding slot remains empty.

### 6.1.1 Generating an English Representation of a Dictionary Entry

The text corresponding to a dictionary entry is generated by means of a simplified version of the Augmented Transition Network (ATN) formalism (Woods

1970, Miller 1983). The ATN is implemented by mutually recursive word-generating and list-generating functions. Let us now describe the generation of text corresponding to a given keyword. First, the keyword is looked up in the dictionary by the word-generating function. If the desired form was specified (e.g., word form or clause form), a representation is randomly selected from the requested slot. Otherwise, it can be selected from either slot. If the chosen representation is a list of words, the list-generating function is activated. Otherwise, if the type of the dictionary entry is non-null (verb or noun), the appropriate derivation is generated.

The list-generating function scans all the words in the list, looking for keywords, and activates the word-generating function for each keyword. The rest of the words in the list remain unchanged.

Once a representation of the entire list has been generated, it is checked against the representation most recently used for the dictionary entry under consideration. If they are equal, an alternative representation has to be produced. The resulting text contains a list of generated words, possibly sprinkled with parameters.

To illustrate the operation of the dictionary, let us go over the generation of a representation of the keyword *new*. The relevant portions of the dictionary are illustrated in Fig. 6.1. For this example we shall assume that the current clause number is 30, and that the dictionary's first selection was the word "new." However, upon verification against the last used slot, the system discovers that this version was recently used. Therefore the clause form is chosen.

Since the selected clause is composed of a list of words, the list-generating function is activated. The first keyword in the list is *never*. Thus, the word-generating

Keyword	Type	Last-used	Word-form	Clause-form
before	nil	(3 before)	{before previously}	{(\$person have (never) (study 1p pp) \$subject (before))}
never	nil	nil	{never not}	
new	nil	(26 new)	{new}	
study	verb	nil	{study see encounter}	

Fig. 6.1: Segment of the Phrasal Dictionary

function is activated. Let us assume that the word “never” was chosen. Next, an English representation of the keyword *study* is generated. After a verb is selected, the verb-generating function is called to obtain the desired derivation (grammatical person “1p” is first person plural, and tense “pp” is present perfect), yielding the form “encountered.” Finally, a representation of *before* is generated. For this dictionary entry, both versions are acceptable, since the word “before” was used a while ago. The output of the dictionary is: “\$person have never encountered \$subject before.” After substituting the parameters, we would get clauses such as “we have never encountered this type of equation before” or “you have never encountered it before.” After the text has been generated, the relevant dictionary entries need to be updated.

### 6.1.2 Updating the Dictionary

At each point in the text-generation process the Sentence Composer checks a newly selected representation of a dictionary entry against its last used slot. Occasionally, the system will discover that this representation was recently used, and that the generation of the text corresponding to the entry in question will have to be repeated. When the representation being generated contains several keywords, this occurrence

creates inconsistencies in the dictionary, since updates performed on the last used slots of some of these keywords are no longer valid. We have considered two ways of dealing with this problem. Given a request for a keyword *K*, in the first alternative, we can withhold updating the dictionary until the entire English representation of *K* has been satisfactorily generated. The second alternative consists of updating each dictionary entry as we go along and record its previous value elsewhere. If at some point in the generation process we discover that a representation coincides with the most recently used one, we can roll back and restart.

Even though both alternatives are seemingly equivalent, the first one will yield erroneous results if we are generating text for a list of words which contains the same keyword more than once. In this case, the word-generating function might come up with the same representation of both instances, since the dictionary was not updated after a representation of the first instance was generated. In addition, the first alternative is somewhat less efficient than the second one, since we have to access the dictionary a second time after the generation process has been completed. For the second alternative, on the other hand, we update each dictionary entry while it is being considered, and we have to access the dictionary again only in the event of failure. Due to the multiplicity of choices, the incidence of this event is rather low. These considerations lead us to choose the second approach.

### **6.1.3 Generating Pronouns**

As stated previously, the rhetorical rules applied in the generation of pronouns are rather straightforward. Nevertheless, the resulting usage of pronouns contributes to the fluency of the text.

When referring to pronouns in the context of the dictionary, we shall use two concepts: the pronouns commonly used in the English Grammar (e.g., it, them, etc) shall be denoted *full pronouns*, whereas references to previously mentioned items by their type (e.g., “this topic,” “those equations,” etc), shall be called *partial pronouns*.

Due to the hierarchical structure of the discourse (see section 3.1.3), we can assume that at each point in time the item in focus can be a topic, an equation, an alternative or a method. The following rules for the generation of pronouns are based on this assumption:

- i. Given an item in focus, if the last reference to this item was not made by means of a pronoun, then the choice between a partial and a full pronoun is random.
- ii. If the item in focus was referred to by means of pronouns more than, say, 4 consecutive times, then no pronoun shall be used, and this item shall be referred to by its entire name.
- iii. A full and a partial pronoun referring to the same item, alternate in the text, as long as the previous rule does not hold.

## 6.2 The Attribute-Clause Generator

This component applies rhetorical rules to generate an English representation of the attributes of a technical utterance. The generated phrases may be of two kinds: dependent and independent.

A dependent phrase is a noun phrase which follows a noun and verb produced earlier

by an Utterance Generator (see section 6.3). For example:

**“Let us consider a very important topic, namely linear equations, which we have not seen for a while.”**

**“We shall devote part of this session to the topic of third-order equations, which, as you already know, is interesting and quite challenging.”**

An independent phrase consists of an entire sentence such as “This method has to be practiced some more.”

The generation of dependent and independent phrases is quite similar, therefore, unless otherwise stated, the process described next is applied for both types of text.

Given the type of a mathematical entity (e.g., topic, rule) and its name (e.g., linear, collect terms), the Attribute-clause Generator extracts its attributes from the complexity and length slots of the extended message in question and the *highlight-attributes* MTU-directive in its motivation slot; and searches for the *new* attribute in the implementation slot. Since there is no causality relationship between these attributes, the selection of connectives depends on rhetorical constraints and the motivation relation between these attributes.

Next, these attributes are separated into two groups: *previously-stated* and *current*. Whereas the former are accompanied by an implementational MTU, the latter are not.

When generating dependent text, the Attribute-clause Generator selects which of the current attributes shall precede the subject of the verb phrase and which shall

follow it. The selection of these attributes is based on two considerations: first, in order to ensure a stylistically sound sentence, the number of attributes which can precede a noun, has to be limited to, say, 3. In addition, only attributes in word form can precede a noun. Among the current attributes which meet these specifications, the attributes which shall precede the noun are randomly selected.

When generating independent text this selection process is not performed and all the current attributes follow the noun. An independent attribute clause with attributes that precede and follow the noun is not generated by the current version of this system.

Once the selection process has been completed, both the current attributes which precede the noun and the current attributes which follow it, are sorted first according to their degree, next according to their motivation relation and finally according to their syntactic form.

i. Syntactic form.

Sorting by the attributes' syntactic form precludes the generation of text containing one clause for each attribute, e.g., "This topic is important, appears quite frequently, is challenging and illustrates some useful techniques," in favour of a representation in which all attributes in word form appear in the same clause. For example, "This topic is important and challenging, appears quite frequently, and illustrates some useful techniques."

ii. Motivation relation.

Sorting by this parameter precludes the generation of a rather confusing clause with several additive and adversative MTUs, in favour of a clause containing

at most two connectives for each syntactic form. For instance, “This method is important **and** useful **but** easy to learn.” Attributes whose motivation relation is null precede attributes of the other two types, and they are connected by means of an additive conjunction.

iii. Degree.

As seen in section 3.1.1, each attribute may be associated with a number which represents its degree. For example, a method can be *very* useful, *rather* useful, etc. By sorting according to this parameter we preclude multiplicity of adverbs, in favour of one adverb for all attributes with the same degree, e.g., “very interesting and challenging topic, and also frequently encountered.”

The effect of sorting the attributes according to these parameters is that all attributes with the same syntactic form appear together, and among those, all attributes with the same motivation relation are generated consecutively, and so on. Thus the sorting process accounts for the generation of attribute clauses like the following:

“An important **and** interesting **and** also rather useful technique.”

In this example, all attributes have the same syntactic form and motivation relation, but the degree of the last attribute differs from the degrees of the first ones.

“A rather important **and** interesting **but** quite simple equation.”

In this case, all attributes have the same degree, however the first two attributes have a motivation relation of type 1, whereas the last one has a motivation relation of type 2.

“The topic of linear equations appears quite frequently, illustrates some useful



methods, and is important and also rather interesting.”

In this text, the first two attributes have a clause syntactic form and the last two have a word form. The additive connective which appears between these types of attributes, namely “and,” is generated by means of a rhetorical rule presented in the next section.

In the next step, the attributes which depict the knowledge status of the student are extracted from the extended message. These are *new* (if not already generated), *known* and *similar* (from the implementation slot), and *increment-knowledge*, *knowledge-preservation* and *practice-reassure* (from the motivation slot).

Finally, the Attribute-clause Generator activates rhetorical rules to compose the rest of the attribute clause from the remaining current, previously-stated and knowledge-status related attributes.

### 6.2.1 Some Heuristics for the Generation of Attribute Clauses

To generate a stylistically sound attribute clause, the following rhetorical rules have been implemented.

- i. All previously-stated attributes shall be presented together and they shall be preceded by an implementational prefix. This prefix is an adverbial phrase which represents the lowest common denominator between the implementational MTUs of the previously-stated attributes. For example, if there is only one such attribute, then it is possible to generate a rather accurate implementational prefix like “As I said in equation number 3.” However, if there are several different previously-stated attributes, the compound implementational MTU is rather vague, e.g., “As we have already seen a couple of times.”

- ii. Previously-stated attributes should follow any other type of attribute or appear in an independent sentence. This is due to the fact that an implementational prefix affects everything that follows it up to the next period. Thus, in order to affect only the attributes which were previously mentioned, they shall either appear last in a dependent clause, or they shall appear in an independent sentence. In order to illustrate this point, let us consider the following sentences:

“Let us examine a method, which, as I said before, is rather complicated and very useful.”

“Let us consider a method, which is very useful, and as I said before, rather complicated.”

Whereas the first sentence stipulates that both attributes have been stated before, according to the second one, only the complexity attribute was presented previously.

- iii. All current attributes shall appear together. This heuristic precludes the generation of a sentence like “This type of equation is very simple, we have seen it before, and it is quite common,” in favour of “We have seen this type of equation before, and it is very simple and also quite common.”
- iv. If there is more than one knowledge-status related attribute, a dependent clause shall include only knowledge-status related attributes, and the rest of the attributes shall appear in a forthcoming independent clause. Otherwise, a knowledge-status related attribute will appear in the same sentence with at least one additional type of attribute. The second part of this rhetorical rule precludes the generation of a sentence which contains only one knowledge

attribute, if attributes of other types have to be generated. In order to illustrate the impact of the first part of this rule, let us consider the following sentences:

“The method of factoring out common factors has already been exercised **but still demands more practice**. This technique is used quite frequently.”

“The method of factoring out common factors has already been exercised, demands more practice, **and** is used quite frequently.”

“The method of factoring out common factors is used quite frequently, has already been exercised **but still demands more practice**.”

The first part of this rule favours the generation of the first example. This is due to the fact that the addition of text corresponding to a current or previously-stated attribute to a clause which already contains two knowledge-status related attributes, tends to cloud their relationship.

- v. If the attributes with syntactic clause form have different non-null motivation relations, and so do the attributes with syntactic word form, then the attributes which appear in clause form and the attributes which appear in word form shall not be presented in the same sentence. This rule precludes the generation of a sentence with two adversative conjunctions.
- vi. If the total number of clauses to be generated does not exceed a certain threshold, say 4, then, in most cases, all attributes shall appear in the same sentence. In such cases, the knowledge-status related attributes shall be followed by the current attributes. As specified in the second rule, the previously-stated attributes are always last. In all other cases, the current attributes and

previously-stated attributes appear in different sentences. This rule is activated after the two previous ones, to determine the organization of the remaining attributes. The following examples illustrate the first and second parts of this rule respectively:

“We shall discuss a topic, namely quadratic equations, which we have not seen for a while, is rather interesting, **and, as I stated before,** is very important.”

“We shall discuss a topic, namely quadratic equations, which we have not seen for a while **and, as I stated before,** is very important. **Furthermore,** it is rather interesting.”

- vii. The following rule is applied to generate a connective between two phrases containing current or previously-stated attributes:

Given two lists of attributes, A and B, if all the attributes in A have the same non-null motivational relation, and no attribute in B has the same motivational relation as the attributes in A, and at least one attribute in B has the opposite motivational relation, then the conjunctive relation between the sentences featuring A and B is adversative. Otherwise it is additive.

This rule accounts for the connectives in the following cases:

“As I have stated before, this type of equation is very important and appears quite frequently. **Notice, however,** that it is easy to solve.”

In this text, the first attribute has motivation relation of type 1, the second attribute has a null motivation relation, and the last one has a

motivation relation of type 2, thus fulfilling the conditions of the rule.

“Let us examine the method of factoring out common factors, which we have not seen for some time and is very important. In addition, as I have said in equation 5, it is rather useful but easy.”

In this example, the first attribute has a null motivation relation, and the second one has a motivation relation of type 1. In the second sentence, the attributes have motivation relations of types 1 and 2 respectively. Thus, the conditions of the rule are not fulfilled, and an additive connective is generated.

These heuristics shed some light on the problem of generation of a certain class of attribute clauses, and enable the system to produce a variety of stylistically sound attribute clauses in the area of algebra.

### 6.3 The Utterance Generators

Human tutors usually know in advance which information they want to transmit about a topic or equation, the number of alternatives they wish to present, the number of steps in each alternative, etc. Inside this framework, they plan their commentary a few sentences at a time. Likewise, given a partially ordered list of extended messages (see Appendix 2), the generators in the Sentence Composer scan predefined subsets of these messages, to determine the manner and order in which these subsets and their corresponding MTUs should be presented.

An additional feature which characterizes human discourse is the proper use of pronouns. The need for a pronoun at any stage of the text-generation process depends on the items which were in focus in previous statements, on the items currently in

focus, and on the presence of intervening statements (Reichman 1978, 1984, Grosz 1977). In order to produce fluent discourse, a generator has to recognize and gather the information that will enable the generation of pronouns when necessary.

The generators in the Sentence Composer form a hierarchical structure. Those which are uppermost in the hierarchy perform high-level decisions regarding the interaction between several technical utterances, whereas those in the leaves confine themselves to one type of extended message (see Fig. 6.2). At the top of the hierarchy, we find the Declarative and the Procedural Generators. Each is activated on a different part of the discourse.

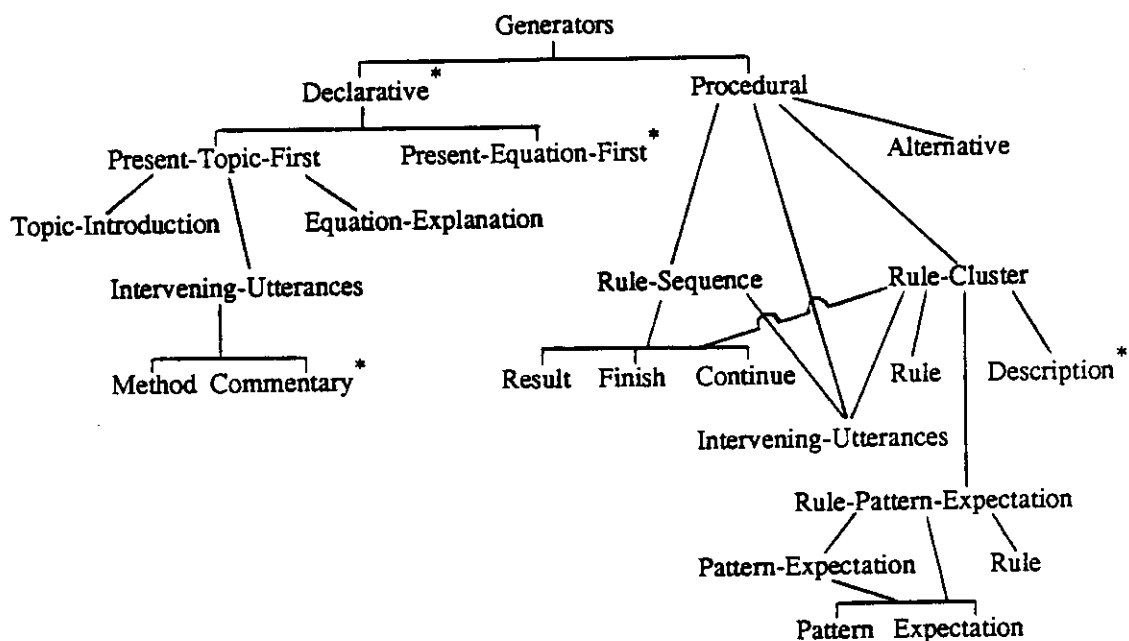
#### Declarative.

The declarative portion of a technical discourse describes the problem under consideration and provides additional background information. The topic, equation and alternatives extended messages belong to this part of the discourse, and intervening utterances may appear in it as well.

#### Procedural.

The procedural part of a technical discourse describes the possible solutions for a given problem. The following technical utterances are part of this group: alternative, rule, pattern, expectation, description, result, finish and continuation. Intervening utterances may be featured in the procedural discourse too, however they may not appear inside a rule cluster, i.e., the technical utterances between a rule and its result (see Appendix 2).

Before we describe in detail the operation of each generator, we shall consider



**Fig. 6.2: Hierarchy of Generators in the Sentence Composer**

some features which are common to the text generated by all of them.

### 6.3.1 Structure of the Generated Text

In general, the text corresponding to a technical utterance is composed of two sections: preliminary and main.

#### Preliminary Section.

This section precedes the rest of the generated utterance, and expresses the relationship between this utterance and expectations and foci established in previous utterances. These relationships are mostly expressed by expectation, affective, digression and focal MTUs.

---

\* These components have not been implemented

Whereas the syntactic form of an affective MTU is based on local aesthetic considerations only, the syntactic form of an expectation MTU may be determined by the Comprehension-Processes Module as well, according to the number of technical utterances elapsed between an expectation statement and its realization or violation (see Chapter 4). The generation process of the expectation, affective and digression MTUs is common to all types of technical utterances, and is guided by the following considerations:

- i. A digression MTU precedes the rest of the sentence.
- ii. If only an affective MTU is required, it is generated in word form.
- iii. A lone expectation MTU is generated according to the specifications provided by the Comprehension-Processes Module.
- iv. If a digression MTU was generated, then both, an affective and an expectation MTU shall have a clause form. This rule precludes the generation of “Anyway, fortunately” in favour of “Anyway, it is rather fortunate that.” If all types of MTUs appear in the extended message, the following text may be generated: “Incidentally, contrary to our expectations, it is rather unfortunate that.”
- v. If a digression MTU is not required, then an affective and an expectation MTU are combined in the following manner. In most cases, the affective MTU shall be generated first, yielding text like “Fortunately, however” or “Unfortunately, contrary to our expectations.” In a less frequent, but still stylistically sound alternative, the expectation MTU would precede the affective MTU, resulting in text such as



“Nevertheless, it is quite fortunate that.”

Even though focal MTUs are also common to several types of technical utterances, their English representation depends on the type of the technical utterance under consideration. For example, the representation of a focus-close MTU-code for a topic technical utterance could be “Let us consider another topic,” whereas the same MTU-code might have the following representation for a rule technical utterance: “After removing parentheses.” In addition, there exist MTU-codes which are featured only in certain technical utterances. For instance, the MTU-code *premature-end* may appear only in topic technical utterances. Both the focus-close and premature-end MTUs appear in the preliminary section of the generated text. However, their generation is performed by dedicated Utterance Generators.

#### Main Section.

This section generates the information pertaining to the current technical utterance. It may include technical knowledge, length, complexity and motivational attributes of the item in focus, and the knowledge status of the student. The English representation of these information items and the manner in which they interact, depend on the type of the technical utterance in question. For example, the *known* implementational MTU-code produces the following text for an equation and a statement, respectively: “This equation is similar to” and “As I said before.” Therefore, the generation of the main part of each technical utterance is performed by a dedicated Utterance Generator.

### 6.3.2 Selection of an English Representation

An MTU-code may have several English representations. The choice of a particular one may be influenced by the position of the corresponding MTU in the generated text and the presence of other technical or meta-technical utterances. For instance, let us consider the MTUs produced for the implementational MTU-code *new*:

1. “We shall now **introduce** the topic of quadratic equations.”
2. “Let us consider the topic of quadratic equations, **which we have not encountered previously.**”
3. “Let us examine a **new** topic, namely quadratic equations.”

### 6.3.3 The Declarative Generator

This component<sup>†</sup> determines the relative order of topic and equation technical utterances. Its decisions are based on the following considerations:

- i. If the number of equations is mentioned, an introductory topic statement shall precede the equation, and information pertaining to the equation shall appear in an explanatory statement following it. The following example illustrates these statements:

“Let us consider a few exercises in the topic of linear equations. We shall begin with the following one:

---

<sup>†</sup> The present implementation does not contain such a generator, the system automatically decides on the generation of introductory topic statements, followed by explanatory equation statements.

$$2x - 3 = 7$$

This equation is quite simple. There is only one way to solve it.”

- ii. If the motivation or complexity slot of the topic extended message is not empty, or the MTU-code *new* appears in its implementation slot, an introductory topic statement shall be generated. This rule is designed to preserve the flow of discourse, by presenting first the information about the uppermost item in the discourse hierarchy, namely the topic, then presenting the information about the next item, i.e., the equation, etc. Thus precluding the generation of a declarative statement like the following:

“Let us now consider the following equation:

$$x^2 - 4 = 0$$

This is a quadratic equation. We have never seen this topic before, and it is very important. There are two ways of solving this equation.”

In favour of the following alternative:

“Let us now consider the important topic of quadratic equations, which we have never seen before. Here is an equation:

$$x^2 - 4 = 0$$

There are two ways of solving it.”

- iii. If intervening utterances appear in the declarative section, then the order of the input is preserved. As seen in Chapters 4 and 5, the presence of intervening utterances may cause the generation of digression MTUs, violation of expectation MTUs, etc. By altering the order in which the extended messages are presented, these MTUs would become incongruous.

- iv. If no topic extended message was generated by the Tutoring Strategist, then an introductory equation statement is mandatory.
- v. In all other cases, the choice between presenting first the information about the topic or the information about the equation is random. However, preference might be given to the generation of an introductory equation statement and an explanatory topic statement, if the motivation, implementation or complexity slots of the equation extended message are not empty. In the following example, the equation is being motivated by obligation:

“I would now like you to solve the following equation:

$$2x - 3 = 7$$

As you probably know, this is a linear equation.”

As a result of the application of these rules, the declarative portion of the text is generated either by means of the Present-topic-first Generator or the Present-equation-first Generator. The former has been implemented, and it activates the Topic-introduction, Intervening-utterances and Equation-explanation Generators, in the following order:

[Intervening-utterances]<sup>†</sup>  
 Topic-introduction  
 [Intervening-utterances]  
 Equation-explanation  
 [Intervening-utterances]

### 6.3.3.1 The Topic-Introduction Generator

When generating an introductory topic statement the Topic-introduction Generator produces first the preliminary part of the statement. To this end, the Topic-

---

<sup>†</sup> The square brackets signal that the intervening-utterances are optional.

introduction Generator begins by checking for the presence of a digression-close MTU-code. If it exists, an MTU like “anyway” is generated. Since no affective or expectation MTUs are associated with a topic technical utterance, next, the Topic-introduction Generator looks for a premature-end MTU-code in the length slot. This MTU signals that the tutor has changed the topic before presenting all the equations he predicted for the previous topic, e.g., “I know I said we would be solving some more linear equations, however I feel we have done enough.” In order to generate it, the dictionary is accessed with the keyword *premature-end*.

Before approaching the generation of the main part, the Topic-introduction Generator looks for MTU-codes which could be produced at this point. This task is performed by the following rules.

If the MTU-code *return* appears in the implementation slot, a sentence like the following could be generated: “Let us **return** to the topic of linear equations.” In addition, if the *knowledge-preservation* MTU-code appears in the motivation slot, then it is combined with the return MTU, yielding a sentence like the following: “Let us **review** a topic which **we have not seen for a while.**”

Alternatively, if the number of equations is to be mentioned, and the focus slot contains a *close* MTU-code, then a clause which closes the topic previously in focus may be generated. For example, “Let us try another topic.”

The format of the main part of a topic statement depends on the number of equations to be presented:

If more than *several* equations are to be mentioned, the main part of the statement will have the following format:

{ *introduction session-part topic-clause length-apology* }

Where:

topic-clause { 1. *dependent-topic-attribute-clause*, or  
2. *action-name topic-name. Independent-topic-attribute-clause*

The following types of sentences are produced by using the first and second formats for the topic-clause, respectively:

“We shall devote most of this section to the topic of linear equations, which is very important.”

“We shall dedicate most of this session to the revision of linear equations. This topic is very important.”

If the number of equations to be presented is less than or equal to *several*, the format of the main part of the statement is:

{ *introduction verb topic-clause* }

Where:

topic-clause { 1. *dependent-topic-attribute-clause*, or  
2. *number-of-equations topic-name.*  
*Independent-topic-attribute-clause*

The first format is selected when the number of equations is not mentioned. The following sentences are produced by the first and second format, respectively:

“We shall now continue with the topic of quadratic equations, which ... .”

“Let us consider several linear equations. This topic ... .”

The *introduction* part may have one of the following representations:

“I would [now] like to.”

This representation is used when the motivation slot contains an obligation MTU-code, transmitted to the student as a wish of the tutor. The brackets enclosing the word “now” signal that it shall be generated, unless an MTU which performs the focus-close function was produced previously.

“We shall [now].”

This representation is generated when the number of equations is to be mentioned and there are more than *several* equations.

“We shall [now]” or “Let us [now].”

One of these introductions is randomly selected, if none of the above conditions is met.

We shall now describe the generation of the rest of an introductory topic statement in which more than *several* equations are presented.

The *session-part* corresponds to text like the following: “devote half of the session,” “dedicate most of the session,” etc.

If *return* or *continue* appears in the implementation slot, or *add* is featured in the length slot, then the session-part is followed by an action-name and a topic-name, yielding text like the following: “We shall devote part of this session to the revision of linear equations,” for the *return* MTU-code, and “We shall dedicate most of this session to practicing **some more** third degree equations,” for the other two MTU-

codes. If these MTU-codes are absent, the main part of the statement shall feature a dependent-topic-attribute-clause, yielding a sentence such as “We shall devote part of this session to an important topic, namely quadratic equations, which demands some more practice.”

If the number of equations is less than or equal to *several*, the generation of the rest of the main part of an introductory topic statement is performed as follows:

The *verb* is determined by the following considerations. If *new* appears in the implementation slot, the verb “introduce” may be used. If *return* appears in this slot, the verb “review” is generated. If *continue* is featured in the implementation slot, or *add* appears in the length slot, the verb “continue-with” is derivated from the dictionary. Finally, if none of these conditions holds, the verb “consider” is used.

The following examples illustrate some introductory topic statements generated by the Topic-introduction Generator:

1. **“Let us now examine the topic of third degree equations, which is interesting and also rather challenging.”**

This text corresponds to an extended message featuring (*close* quadratic) in the focus slot, and two attributes in the motivation slot.

2. **“Anyway, we shall continue with a few linear equations.”**

This example is generated from an extended message containing *close* in the digression slot, *continue* in the implementation slot, and (*mention* 2) in the length slot.

3. **“Let us now review a topic which we have not seen for a while: We shall**



**devote most of this session to equations with fractions. I know these are plenty of exercises, but it is necessary.'**

This text is generated from an extended message containing the following values: (*close third-degree*) in the focus slot, *return* in the implementation slot, *knowledge-preservation* in the motivation slot and (*mention 6*) and (*apologize length*) in the length slot.

4. **"Let us try another topic. We shall dedicate most of the session to the revision of equations with fractions. This topic has not been practiced for some time. I know these are a lot of equations, but it is necessary."**

This text is generated from the same extended message as the text in the previous example. Due to some random choices performed by the Topic-introduction Generator, in this example the focus-close MTU is presented in the first sentence. This causes the implementational MTU to be generated in the main statement. Finally, since the motivation was not generated before the main part, an independent-attribute-clause containing it has to be added.

### 6.3.3.2 The Equation-Explanation Generator

This component produces an explanatory statement corresponding to an equation technical utterance. This type of statement is divided into two parts.

Equation Introduction.

This part is rather brief and consists of an introductory sentence such as:

"Here is an equation" or "Let us consider a rather simple equation."

Equation Commentary.

This part follows the given equation, and presents the information extracted

from the equation extended message, and possibly from an alternatives extended message. For example, "This equation is very difficult. There are two alternatives for solving it."

The Equation-explanation Generator first generates the equation introduction part. In order to accomplish this task, it first generates an MTU which closes a digression, e.g., "anyway." Next, the Equation-explanation Generator decides whether a pronoun is required. The use of a pronoun in the equation introduction part depends on the information stated in the immediately preceding statements. If the word "equation" was mentioned, and no intervening utterances or independent clauses were generated thereafter, then a pronoun has to be generated. A condition which ensures that the word "equation" indeed was previously stated, is that the number of equations introduced in the topic-related statement was no greater than *several* (see section 6.3.3.1).

After the need for a pronoun or lack thereof has been established, the contents of the equation-introduction part are determined by means of the following rules.

- i. If only one equation was previously introduced, then, if a pronoun is required, the equation introduction part shall be "Here it ~~is~~." Otherwise (a pronoun can not be used), the equation-introduction part could be "The equation follows."
- ii. If the number of equations was not introduced, then the equation introduction part would be "Here is an equation." Notice the use of an undetermined article if the word "equation" was not mentioned before. If the equation does not have a distinguished pattern and it is not similar to a previous equation, the complexity attribute may appear in the equation introduction part. In this case,

an introductory statement like the following could be generated: "Here is a rather simple equation."

- iii. Finally, if more than one equation were presented in the topic introduction, the equation extended message would require a temporal MTU, yielding an introductory sentence like "Let us begin with the following equation," if a pronoun can not be used, and a sentence like "We shall begin with the following one," otherwise. As before, if the complexity attribute is chosen to appear in the equation introduction part, the following sentence could be generated: "We shall start with this very difficult exercise."

Once the equation-introduction part and the equation itself have been presented, the Equation-explanation Generator proceeds to generate the equation-commentary:

First, if a *complexity-reassure* MTU-code appears in the extended message, then a statement to this effect is generated. This statement could have one of the following formats:

*complexity-reassure*

This type of sentence is produced only if the complexity attribute of the equation was presented before the equation. The following sentence is of this type: "Do not worry, since I shall explain its solution a couple of times."

*[implementational MTU] complexity complexity-reassure*

This type of sentence is generated if the complexity attribute did not precede the equation. The following statement illustrates this format: "This equation is very difficult. Nevertheless, you should not be concerned, since we shall go

over its solution a few times.’’ If an implementational MTU is required, and there are no MTU-directives in the motivation slot of the extended message, the implementational MTU should also be represented in the current statement. This policy precludes the generation of the following text: ‘‘This equation is very difficult. Nevertheless, you should not be concerned, as we shall go over its solution a few times. It is similar to equations 1 and 3,’’ in favour of: ‘‘This equation is similar to equations 1 and 3 and is very difficult. Nevertheless, you should not be concerned, as we shall go over its solution a few times.’’

Next, the system produces an independent attribute-clause which includes the remaining MTU-s in the implementation, motivation and complexity slots. Notice that if a complexity-related MTU were included in the extended message, without an accompanying *complexity-reassure* MTU, the corresponding complexity attribute would appear in this independent clause.

As stated in section 4.5, a given equation may also be motivated through the methods applied to solve it. The MTU-codes for this type of motivation appear in an *alternatives* extended message. The following motivational statements are based on these MTUs:

1. ‘‘This equation enables us to practice a few methods, which we have not seen for some time.’’  
This sentence is generated when more than one method require a *knowledge-preservation* motivation.
2. ‘‘Through it we can exercise the very useful technique of removing

parentheses, which demands some more practice.”

This type of sentence is generated when the equation under consideration is a vehicle for practicing one particular method.

3. “By means of this equation we can introduce an important technique, namely factoring out common factors, but first, let us consider two other ways of solving this equation, for comparison purposes.”

This type of motivation is produced if an equation is presented as a vehicle for introducing a new method, but this method is not featured in the first solution alternative. In this case, information in the length slot of the equation extended message is ignored, as it is subsumed by the information in the generated motivational statement.

Finally, a statement which introduces the alternatives that solve a given equation is generated. This type of statement is extracted from the length slot of the equation extended message. If all the alternatives shall be mentioned, the MTU-code in the length slot refers to the number of *existing* solution alternatives, otherwise it refers to the number of *mentioned* alternatives. In the first case, a sentence like the following could be generated: “There are three ways to solve this equation,” whereas in the second case the following sentence might be produced: “We shall consider a few alternatives for solving it.” If an *apologize-length* MTU-code appears in the length slot, then a statement to this effect is generated thereafter.

This concludes the generation of the declarative portion of the text. The generation of intervening utterances shall be presented later on, since they may appear almost anywhere in the discourse.

### 6.3.4 The Procedural Generator

The Procedural Generator first activates the Alternative Generator in order to introduce the solution alternative under consideration. It then scans the subsequent extended messages, to determine whether a given sequence of rules shall appear in one sentence, or each rule cluster shall entail the generation of one or more sentences, requiring the activation of the Rule-Sequence Generator and Rule-Cluster Generator, respectively. This decision is based on the succinctness of the prospective English representation of the rule clusters, and is performed by the following rhetorical rule:

If two or more rule extended messages are presented consecutively, i.e., without rule-dependent or intervening technical utterances between them, all have empty *rule slots* (see Appendix 2), all the rules but the first one have at most a temporal MTU, and the first rule has at most temporal, digression and focus-close MTUs, then these rules are presented in the same sentence.

The following examples illustrate this situation:

“We solve this exercise by removing parentheses, collecting terms and applying the quadratic formula.”

This sentence is generated when all the rules in an alternative conform to the above given criteria.

“Next, we collect terms and complete the square.”

This sentence is generated if the rules in the list appear between other rules.

In the rest of the cases, each rule cluster requires the generation of one or more sentences.

#### 6.3.4.1 The Alternative Generator

The Alternative Generator generates text which introduces a particular approach for solving an equation.

First, the system generates the preliminary part, possibly containing digression, expectation and affective MTUs. This task is performed according to the guidelines presented in section 6.3.1.

Next, an introductory statement is generated. Its format is determined by the following rules.

- i. If an affective MTU-code appears in the extended message, an existential type of statement has to be generated, e.g., "Fortunately, however, there is a way to solve this equation." The rationale behind this rule is that the presence of an affective MTU-code indicates a current belief in the certainty of a solution, after an utterance stating otherwise was issued. Thus, it would be incongruous to say: "Fortunately, let us consider the following alternative."
- ii. If the success of the alternative in question is uncertain, and this fact is consistent with other MTU-codes in the extended message, the alternative can be introduced by means of text like the following: "Let us try another approach."
- iii. If a focus-close MTU-code appears in the extended message, then an introductory statement like the following is generated: "Let us now consider the last alternative" or "We shall now discuss a different approach." Where the object of the statement is dictated by the number of the temporal MTU and its

type, e.g., *last* or (*2 cardinal*), respectively. If no additional MTUs need to be generated, then an introductory statement like the following is produced: “We shall now consider the last solution to this equation, which consists of the following steps.” Otherwise, if only the number of operations to be performed remains to be stated, a dependent clause to this effect is appended to the introductory statement. The following text illustrates the result of this action: “Let us now examine the second alternative, which entails several operations.” Finally, if other MTUs need to be generated, they shall be presented in an independent clause.

- iv. If no focus-close MTU is required, then a policy similar to the one presented in the previous rule is followed, yielding a sentence such as “The first alternative involves the following steps.” If any additional MTU-codes appear in the extended message, no introductory statement is generated, and the information corresponding to these MTU-codes shall be generated in an independent clause.
- v. The first alternative requires special attention. If only one alternative is discussed, this fact was previously advertised, and no intervening utterances were issued, a pronoun should be generated instead of an explicit reference. This rule produces the second sentence in the following example: “There is only one way to solve this equation. It entails many operations, and here it is.” Where the first sentence was generated by the Equation-explanation Generator. If, however, an intervening utterance was generated, then a pronoun can not be used, and the following introductory statement would be generated: “There is only one way to solve this equation. Incidentally, ... . Anyway, here is the



**solution.”**

After the generation of an introductory statement has been completed, the Alternative Generator proceeds to generate one or more independent clauses which contain motivational and length-related MTUs. The following statements illustrate some motivational MTUs.

1. “Through the following solution we shall introduce a very useful and important technique, namely Patt’s guessing method for solving quadratic equations.”

This MTU motivates the student by highlighting the attributes of a new method.

2. “It enables us to exercise a few methods which demand some more practice.”

This MTU motivates the student by means of an increment-knowledge motivation which applies to a few rules.

3. “This alternative also enables us to go over a solution you might have considered.”

This MTU provides an English representation of the *attempted* MTU-code. The word “also” indicates that previous alternatives were similarly motivated.

The alternative statement is completed by generating a length-related MTU and combining it with the previously produced motivational MTU. This task is accomplished by means of the following rules.

- i. If a *length-apology* MTU is required, then a statement like the following is

generated: "This alternative requires many steps, however it has to be considered." If a motivational MTU was generated, it is issued after this sentence.

- ii. If a length-related and a consolatory MTU are required for the existing number of steps, and the number of steps to be mentioned is acceptable to the student, then the resulting text would be: "This alternative requires plenty of steps, however we shall mention only a few of them." Like in the previous rule, a motivational MTU may follow this sentence.
- iii. If both a motivational and a length-related MTU are required, and all the operations are being mentioned, then the length-related MTU shall precede the motivational MTU. Notice that the conjunctive relation between these two items of information is adversative, since the length-related information may be discouraging, whereas the motivating information is generated to encourage the student to attend to the forthcoming explanation. This rule may yield an independent clause like the following: "This alternative involves many operations, however, it enables us to practice a couple of techniques, which we have not seen for some time."
- iv. Finally, if an existential introductory statement was previously generated (due to the presence of an affective MTU-code), and only a length-related MTU is required, the latter shall follow the introductory statement, and like in the previous rule, they shall be connected by an adversative conjunction. The resulting statement would be: "Fortunately, there is a way to solve this equation, but it requires many operations."

#### 6.3.4.2 The Rule-Sequence Generator

This component produces a sentence containing a sequence of algebraic rules. The following rhetorical rules determine whether a temporal MTU needs to be generated, and what shall be its type.

- i. If the alternative under consideration does not contain any rules besides the rules in the sequence, no temporal MTUs are issued, and the solution of the equation is presented by a sentence like the following: “We solve this equation by collecting terms, factoring out common factors and solving separately for each factor.”
- ii. If a focus-close MTU-code is featured in the first rule of the sequence, no temporal MTU shall be issued. This rule precludes the generation of the following sentence: “After collecting terms, next, we transfer a constant to the right hand side of the equation.”
- iii. If the temporal MTU of the first rule in the rule sequence is greater than 2, and this sequence completes the solution of the equation, then an MTU corresponding to the MTU-code “last” is generated. This rule may produce the following text: “We complete the solution of the equation by collecting terms and applying the quadratic formula.”
- iv. Otherwise, the temporal MTU of the first rule precedes the entire sequence, yielding a sentence like: “Next, we remove parentheses and collect terms.”

After the English representation of the temporal MTU has been decided, the following rhetorical rules are activated to determine the format of the rest of the sen-

tence.

- i. If a digression-close MTU-code appears in the first rule in the sequence, then a digression MTU precedes the sequence of rules.
- ii. If a focus-close MTU-code appears in the first rule, its English representation is generated after a digression-close MTU, and present tense shall be used for the verbs representing the forthcoming rules. This rule yields a sentence such as **“In any event, after removing parentheses, we collect terms and apply the quadratic formula.”**
- iii. If no focus-close MTU-code is required, the Rule-sequence Generator randomly chooses between a word form for the temporal MTU (e.g., “next,” “finally”) or a clause form (e.g., “we continue by,” “we complete the solution of the equation by”), where the choice leans towards the more commonly used word form. The tense of the verbs is determined by the selected format, i.e., the word form entails present tense, whereas the clause form requires present participle. This policy yields the following sentences respectively: **“Thereafter, we remove parentheses and collect terms”** and **“We complete the solution of the equation by transferring the constant to the right and side, and dividing by the coefficient of x.”**
- iv. If a result technical utterance follows the rule sequence, a dependent result statement is generated, unless otherwise determined by Result Generator (see section 6.3.4.7). This rule produces the following text: **“We transfer terms to the left hand side of the equation and collect terms, yielding the following result.”**

### 6.3.4.3 The Rule-Cluster Generator

The presence of certain MTU-codes affect both the generation of other MTUs and the order in which the extended messages and MTUs in a rule cluster are generated. The Rule-cluster Generator applies the following rhetorical rules to determine the manner and order in which the information items in a rule cluster shall be presented.

- i. If a pattern or expectation extended message contains a non-null implementation slot, then an MTU corresponding to the *known* MTU-code in the rule under consideration shall not be generated. The rationale supporting this rule is that if the pattern or expectation of a rule is known, so is the rule, and stating this fact separately would be redundant. For instance, this rhetorical rule would avoid generating the second sentence in the following text: “As you know, since 3 is a factor common to the second and third terms, we factor it out. **We have seen this method before.**”
- ii. Affective and expectation-violation MTUs can not be mentioned in conjunction with event-violation MTUs. Therefore if any of the first two MTUs has to be generated, the last one is ignored. This rule precludes the generation of text like: “Fortunately, but first ... .”
- iii. If an event-violation MTU-code appears in a rule technical utterance, then the statement describing the activation of this rule precedes the text representing its dependent technical utterances, namely expectation and pattern. In addition, the presence of an event-violation MTU-code precludes the generation of a temporal and a focus-close MTU. This rule may yield the following sentence:

- “But before we apply the above mentioned technique, we transfer the first and second terms on the left hand side of the equation to the right hand side.”
- iv. A focus-close MTU-code precludes the generation of a temporal MTU, and causes the text corresponding to the rule to be generated before its pattern and expectation. This results in text like the following: “After removing parentheses, we collect the terms on the left hand side.”
  - v. The presence of an expectation or digression MTU requires the generation of a temporal MTU in clause form. This rule would generate “Anyway, we go on by ... ” instead of “Anyway, next, ... .”
  - vi. If any descriptive information, e.g., description extended message, motivational MTUs, etc, has to accompany the presentation of an algebraic rule, then this rule, its pattern and its expectation are stated first, followed by one or more independent clauses, which contain the descriptive information. An independent clause corresponding to the result extended message is generated last. The following text could be generated by this rhetorical rule: “Since both sides of the equation are multiplied by 3, we divide by this factor. As I have stated before, this method is very simple but important. It yields the following result ... .”
  - vii. Finally, if either the pattern or the expectation of a rule are to be presented, then the result of the rule application shall be generated in an independent clause. Otherwise, the result of a rule application appears in a dependent clause, unless the Result Generator decides differently. The following output is generated by the first and second part of this rule, respectively:

“We remove parentheses in order to get simple terms. This operation arrives at the following result.”

“We remove parentheses, getting ... .”

According to the outcome of these rules, the Rule-cluster Generator activates one of the generators in the next level of the hierarchy, to determine the manner and order in which the remaining technical and meta-technical utterances in the rule cluster shall be generated.

If a statement corresponding to the rule activation was not generated by the Rule-cluster Generator, then the Rule-pattern-expectation Generator produces the text corresponding to a rule, its pattern and its expectations by applying the following rhetorical rules.

- i. If both the pattern and the expectation extended message have an implementational MTU, then a sentence of the following format is generated:

*{ implementational-MTU pattern rule expectation }*

An instance of the resulting text is: “As you know, since  $x-1$  is a factor common to the first and second terms on the left hand side, we factor it out, hoping to get a factor common to the rest of the terms.” In general, when presenting the pattern of a rule, a tutor is instantiating a more general pattern. Since FIGMENT does not have memory of exact statements issued, it has to confine itself to implementational MTUs such as “as you already know” as opposed to “as I have stated before.”

- ii. In most cases, a rule shall be preceded by its pattern and followed by its

expectation. If the pattern extended message contains an implementational MTU-code and the expectation extended message does not, then the expectation statement has to appear in an independent clause. Otherwise, the reader will understand that the implementational MTU affects the expectation statement as well. If, on the other hand, the pattern extended message has an empty implementation slot, then, unless otherwise determined by the Expectation Generator, a dependent clause shall be generated for the expectation statement. The following output is generated by the first and second parts of this rule, respectively:

“As you already know, **since** all the terms on the right hand side of the equation are simple terms, we collect them. We perform this step in order to get a more compact expression.”

“**Since** all the terms on the right hand side of the equation are simple terms we collect them, **in order** to get a more compact expression.”

- iii. In some cases the rule statement is generated first, and its pattern and expectation are produced thereafter by means of the Pattern-expectation Generator.

The Pattern-expectation Generator applies the following rules to decide on the relative order between the pattern and expectation statements.

- i. If both the pattern and the expectation extended messages have a non-null implementation slot, then an independent clause of the following format is produced:

*{ implementational-MTU pattern and expectation }*



The following text illustrates the output of this rule: “As you already know, it is possible to perform this operation, because  $x^2$  is a factor common to both terms, and we hope to get a factor common with the rest of the equation.”

- ii. If the pattern extended message has a non-null implementation slot and the implementation slot of the expectation extended message is empty, then an independent clause shall be generated. If the implementation slot of the pattern extended message is null, a dependent clause is produced. This rule precludes the generation of a sentence like the following: “Next, we collect terms, since, as you know, the terms on the right hand side are simple terms,” in favour of “Next, we collect terms. As you know, we are able to perform this operation, because the terms on the right hand side are simple terms.” Notice that these sentences do not have the same meaning. While in the first one the implementational MTU conveys that the fact that is known to the user is that there are simple terms on the right hand side of the equation, the second sentence states that the pattern of an algebraic rule is known. If an expectation statement is required as well, an independent clause shall be generated following the pattern statement.
- iii. Finally, if a pattern extended message is not featured in the rule cluster, but an expectation extended message is, then unless otherwise determined by the Expectation Generator, the expectation statement will have a dependent clause form.

After the relative order of the extended messages has been determined, the text is generated by activating lower-level Utterance Generators.

#### 6.3.4.4 The Rule Generator

When the Rule Generator is activated, only a few MTUs remain to be determined. These MTUs are featured in the following sentences:

**“Nevertheless, we can still remove parentheses.”**

This type of sentence is generated if the rule extended message contains an expectation-violation MTU-code.

**“We continue by collecting terms.”**

If the previous condition does not hold, and a temporal MTU was not generated before, then a temporal MTU in clause form is produced, yielding this type of sentence.

If none of these MTUs are required, a rule statement like the following is produced: “You factor out  $x$  from the first and second terms.” Notice that if a pattern statement precedes the rule, a pronoun will have to be generated, instead of an explicit reference. In this case, the text generated for the last example would be: “Since  $x$  is a factor common to the first and second terms, we factor it out.” If additional pronouns are required, they shall be partial.

Finally, the Rule Generator produces an implementational MTU pertaining to previous applications of the rule under similar circumstances. This results in sentences like the following: “We factor out  $x$  again,” “We can still collect terms, like in previous equations” or “You begin by removing parentheses, like in the second alternative.”

#### 6.3.4.5 The Pattern Generator

The Pattern Generator produces several types of sentences, according to the following considerations.

- i. A pattern extended message can have an implementational MTU-code only. In order to avoid generating misleading text, this type of MTU entails the generation of an independent pattern clause.
- ii. If an independent clause is required, and the rule has already been presented, then a sentence like the following is produced: **“It is possible to apply this method, because  $3x$  is a factor common to the first and second term.”**
- iii. Otherwise, if an independent clause is required and the rule has not been stated yet, the Pattern Generator might generate the first part of the following sentences: **“The terms on the left hand side are simple terms, therefore we can collect them”** or **“Since 4 multiplies both sides of the equation, we can divide these sides by it.”**
- iv. Finally, if none of these conditions holds, a dependent clause is generated, yielding text like the following: **“We factor out  $x^2$ , because it is a factor common to the terms on the left hand side.”**

Since the pattern and rule statements are closely related, the generation of pronouns to reference the same items is essential. This issue is resolved by means of a rather simple rhetorical rule:

A full pronoun is generated for an item featured in the *actor slot*<sup>†</sup> of the technical part of a pattern extended message, and a partial pronoun is produced for any other items, if necessary.

This rule is applicable if a pattern statement precedes a rule statement and vice versa, yielding sentences like the following: “Since  $x^2$  is a factor common to the first and second terms, we factor it out from these terms.”

#### 6.3.4.6 The Expectation Generator

The generation of an expectation extended message is performed by applying the following rhetorical rules.

- i. If the expected event has a probability of 1, i.e., it represents the purpose of an action, an MTU like “in order to” is generated, otherwise the MTU “hoping to” is produced.
- ii. If an independent expectation clause is required, and the rule has already been stated, then a sentence like the following is produced: “**We perform this operation hoping to get rid of terms of third degree.**”
- iii. Otherwise, if a dependent expectation clause has to be generated, a sentence like the following is issued: “We factor  $x$  out, **in order to get a factor common to other terms in the equation.**”
- iv. Finally, a human tutor may also generate an independent expectation clause, if the rule has not been stated. The following sentence represents this type of text: “**We would like to eliminate terms of third degree, to this end we**

---

<sup>†</sup> See Appendices 2 and 3.

remove parentheses.” However, this format can be used only if the object of the expectation statement is independent of the rule and pattern statements.

#### 6.3.4.7 The Result Generator

The Result Generator produces an independent clause, either if it is so specified by the Rule-sequence or Rule-cluster Generators, or if the result statement requires some introductory MTUs. The following sentences illustrate the output of this component:

1. **“Unfortunately, contrary to our expectations, this operations yield the following result.”**

This independent clause is generated from an extended message featuring an expectation-violation and a negative affect-related MTU-code. The decision to generate an expectation-violation MTU in clause form, is based on the position of the result technical utterance relative to the expectation technical utterance. If the result technical utterance appears immediately after the expectation technical utterance, the following text would be generated: **“We remove parentheses and collect terms hoping to eliminate third degree terms. Unfortunately, however, we get ... .”**

2. **“After performing the last operation, we get the result we were hoping for.”**

This independent clause represents a result extended message containing a focus-close and an expectation-realization MTU-code.

3. **“We collect terms, arriving at the desired result.”**

Since no introductory MTUs are required, unless otherwise specified, the

Result Generator produces this type of dependent clause for a result extended message containing an expectation-realization MTU-code.

#### 6.3.4.8 The Finish Generator

This component of the Sentence Composer determines the text to be generated to signal the conclusion of a solution alternative. If the alternative is successful and no MTUs are required, no text shall be generated. Otherwise, if the finish extended message contains focus, digression and affective MTU-codes, the MTUs corresponding to the last two MTU-codes are generated first, yielding sentences such as:

1. **“Anyway, after performing the last operation the equation is solved.”**

This sentence is generated if a digression-close and a focus-close MTU are required.

2. **“Anyway, the above mentioned approach solves the equation.”**

This sentence is produced if a digression-close MTU appears in the extended message.

3. **“Unfortunately, there is nothing else we can do here.”**

✍ This sentence pertains to an alternative which did not succeed in solving the equation, and whose finish extended message is accompanied by a negative affective MTU.

#### 6.3.4.9 The Continue Generator

The generation of text representing a continue extended message resembles the generation of a finish extended message. The following sentences may be produced by this generator:

1. “After performing the last operation, we complete the solution of the equation by removing parentheses and collecting terms.”

This type of sentence is generated if the operations which are not presented still have to be mentioned, and a focus-close MTU-code is required.

2. “From this point you can reach the solution by yourself.”

This sentence represents a continue extended message which requires no mention of the remaining algebraic operations, and does not need any MTUs.

### 6.3.5 The Intervening-Utterances Generator

This generator produces text which describes the methods used to solve certain types of equations, and also general commentaries. These technical utterances may appear virtually at any point in an explanation, and may interrupt the flow of the discourse.

The Intervening-utterances Generator scans the list of extended messages for consecutive appearances of these utterances. For each utterance it activates the appropriate lower-level generator, namely a Method Generator or a Commentary Generator<sup>†</sup>.

#### 6.3.5.1 The Method Generator

The Method Generator produces one or more statements containing the methods available to solve equations of a certain type. The system generates two kinds of method statements:

Existential — In which the existence of a method or lack thereof is mentioned; and

---

<sup>†</sup> The Commentary Generator has not been implemented.

Enumerative — In which the actual names of the methods are mentioned.

In addition, each of these statements may either refer to general methods, which solve all instances of a given class of equations, or specific methods, which are applicable only under certain circumstances.

The following rhetorical rules are applied in order to generate a method statement.

- i. If the MTU-code *also* appears in the implementation slot of the extended message, signaling that the previous extended message contains a method technical utterance too, and if the current extended message has no expectation-violation or digression MTU-codes, then an additive conjunction is generated between the current and the previous method statement. Otherwise, the MTU-codes featured in the preliminary section of the extended message are generated as explained in section 6.3.1.
- ii. If two method statements regarding the same topic or type of equation, are presented consecutively, and both have non-null implementation slots, then a common implementational prefix is generated for both of them.
- iii. If the digression slot of a method extended message is non-null, then since the object of the method statement is not equal to the item currently in focus, it can not be referenced by means of a pronoun, and has to be explicitly stated. Otherwise, a partial pronoun may be generated, unless the Pronoun Generator decides differently (see section 6.1.3).
- iv. A statement regarding general solution methods may be formulated as follows:



“**In general**, we can solve linear equations by ... ” or “There is no **general method** to solve ... .” Whereas statements referring to specific solution methods may have the following representations: “We can solve **certain** linear equations by ... ” or “There are no **particular methods** to solve ... .”

By means of these rules, the Method Generator produces the following text:

1. “**Unfortunately**, we shall not go over a **general** method for solving this type of equations.”

This sentence is generated from a method extended message referring to a general method, whose affective slot contains a *negative* MTU-code and whose technical part has the value *not-teach*.

2. “**Fortunately, however**, there are a few **particular** methods for solving third degree equations.”

The specific method extended message corresponding to this existential sentence has an expectation-violation and a positive affect-related MTU-code.

3. “**In general**, we can solve linear equations by removing parentheses, collecting terms and dividing by a constant.”

This enumerative sentence represents a method extended message for a general method.

## CHAPTER 7

### Conclusions and Future Research

In this research we have modeled Meta-Technical Utterances in terms of their anticipated effect on the listener comprehension, and have incorporated these models in a text-generation facility as tools for generating fluent and cogent discourse.

We have classified MTUs according to their function, as seen by the speaker, in transmitting the subject matter, thus distinguishing between Knowledge-Organization, Knowledge-Acquisition, and Affect-Maintenance MTUs.

Based on this taxonomy, we have designed and implemented the text generation components of an Intelligent Tutoring System called FIGMENT. Specifically, we have demonstrated the generation of Knowledge-Organization and affect-transference MTUs from the structure of the knowledge to be transmitted, and the generation of Knowledge-Acquisition and consolatory MTUs by consulting simplified models of the comprehension processes activated by the listener. We have shown that a text generation process based on such models would capture important rhetorical features that support natural discourse.

#### 7.1 Limitations of this Research and Future Work

The system implemented in this research generates believable text, i.e., comparable in many respects to discourse produced by a human tutor. As a next step, it is important to test the transferability of this facility to alternative domains. This

depends primarily on the adaptability of the Comprehension-Processes Module. Most of the processes activated by the Comprehension-Processes Module rely on the hierarchical problem-solving structure of the transferred knowledge. This property is shared by many technical tutoring domains. The extensibility of the Comprehension-Processes Module to these domains hinges on its ability to incorporate new types of technical utterances, since the presence of a technical utterance or its accompanying MTUs may influence the need for MTUs in subsequent utterances.

The development of FIGMENT has opened issues which are basic for the enhancement of the quality of machine-generated text.

- i. The design proposed in this work distinguishes between the strategic and the tactical components of the system. However, like the generation of MTUs, the choice of material to be discussed may also be influenced by affect-related considerations. Therefore, it would be profitable to explore a text-generation scheme in which both, the subject matter and the MTUs are produced at the same time, by activating processes similar to the ones applied by the Comprehension-Processes Module.
- ii. A tutor often qualifies estimational and attribute-related MTUs by means of short-term-implementational or adversative MTUs, yielding sentences like "This method **also** requires several computations" or "Unlike the previous **equation**, this exercise is easy," respectively. The presence of the qualifying MTUs lends an impression of intelligence on the part of the speaker, in that he exhibits recollection of previous discourse. To generate these MTUs, a system should use a model of the listener's short-term memory, and a set of rhetorical rules that weave these MTUs into the text.

- iii. FIGMENT generates a particular Knowledge-Acquisition MTU based only on the affects evoked by a technical-utterance. The non-technical goals of the listener are not explicitly taken into consideration. For example, an estimational MTU like “this technique is quite difficult” may imperil a student’s goal of quickly mastering the method in question, and might require the generation of a consolatory MTU. At present, the generation of the latter MTU is not goal-based, but reflects only the disparity between the capabilities of the student and the difficulty and/or length of the technical-utterance in question. It is our contention that it is worth while incorporating and consulting an explicit model of the student’s goals.
- iv. The expectations modeled by FIGMENT concern only the solution of an equation. However, during a tutorial session a student experiences other expectations, e.g., he may expect a particular topic to be discussed or the forthcoming equation to be of a certain difficulty, he may be tired and wish the session to be finished, etc. The latter case should prompt a sentence such as: “Let us solve just one more equation.” The incorporation of a richer model of the listener’s expectations will improve the quality of the generated discourse.
- v. A text-generation system should be able to present a *relative* evaluation of the entities under consideration, generating a sentence such as “This technique is more efficient than the one used in the last alternative.” Currently FIGMENT presents only absolute attributes of the mathematical entities in question.

## APPENDIX 1

### Taxonomy of Conjunctions by Wilkins

This section contains an excerpt of the taxonomy of the English Language performed by Wilkins (1668):

"Conjunctions are such Particles as serve for the *joyning* together of *words*, or rather of *sentences*. Of these there may be reckoned these four Combinations or twelve paire; though all of them be not alike simple and of equal necessity, yet there is none of them without its particular convenience.

*Interrogative, // Affirmative, or Negative.*

WHETHER YEA?

WHETHER NO?

*Conjunctive // Affirmative, or Negative,*

AND

NEITHER

*Conditional // Affirmative, or Negative,*

IF, so that

UNLESS.

*Approbative, or Discretive and restrictive,*

INDEED

BUT

*Concessive or Exceptive*

ALTHOUGH

YET

*Disjunctive // Definite, or Indefinite,*

OR

EITHER.

The third combination are all of them *Causal*

*Adjunctive of the end; whether cause or Event; // Affirmative or Negative,*

THAT, to the end that,

LEAST THAT

*Ratiocinative, belonging to the Antecedent; whether // that which makes if follow the consequent: or that which may indifferently precede or follow.*

FOR

BECAUSE

*Ratiocinative belonging to the Consequent; whether // interrogative and indefinite: or illative, and demonstrative,*

WHY, wherefore, *what is the cause or reason,*

THEREFORE.

*Declarative; whether // of the cause or of the event,*

WHEREAS, *seeing that, sith that,*

THEREUPON

*Additional, and transitional, whether // continuative, or suppletive,*

LIKEWISE, *also, together with, moreover,*

AND SO FORTH, &c.

*Expositive*; either || by *Synonyme*, or by *Instance*,

TO WIT, *viz.*

FOR EXAMPLE, EXGR.

The three last of these are not properly Particles or single words, but rather the Contractions of several words, they are here added to the rest for greater convenience, partly for compleating the number and filling up the vacancies; and partly in Compliance with the use of most vulgar Languages, when they write contractedly."

## APPENDIX 2

### Output of the Tutoring Strategist

The input to the tactical components of FIGMENT is a list of technical utterances (TUs) which have the following format:

*((technical-information) processing-information)*

The technical-information part contains information pertaining to the subject in question, e.g., the name of the topic, the operation to be applied, etc. Unless directives to the contrary have been provided, this information should be transmitted to the student. The processing-information part contains data required for the text generation task.

The TUs appear in a partially predefined sequence, which is depicted in Fig. 2app.1. We recognize three types of TUs: Independent, Rule-dependent and Intervening.

#### Independent.

These TUs describe the topic and equation under consideration (*topic* and *equation*) and the actions taken to solve it. They signal the beginning of a solution alternative (*alternative*) and its termination (*continue* or *finish*), and present the algebraic operations performed to solve the equation (*rule*).



### Rule-dependent.

These TUs are optional and may only appear after a rule TU. They pertain to the rationale of applying a rule (*pattern*), the expectations it triggers (*expectation*), its mode of application (*description*) and its result (*result*). The relative order of the rule-dependent TUs is immaterial, and together with the preceding rule they form a *rule cluster*.

### Intervening.

These TUs are statements, which can appear anywhere in the TU-list, except inside a rule cluster. The TUs included in this type are *method-statements* and *commentaries*.

In a fully implemented Tutoring System, the list of TUs should be generated by the Tutoring Strategies Module, however in the present system it is hand-coded.

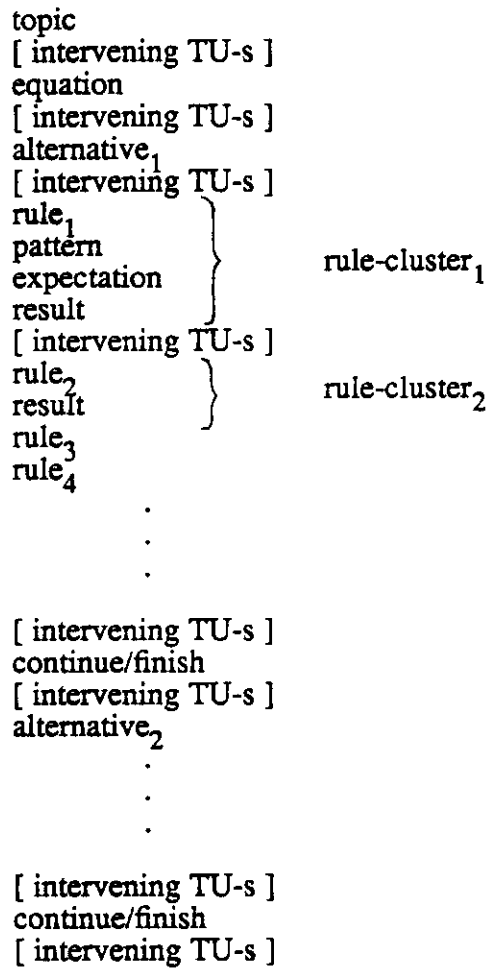
The following technical utterances are representative of the domain of algebra.

((*topic topic-name*) *mention*).

This utterance contains the name of the topic to which the forthcoming equation belongs (e.g., linear, quadratic, third degree, etc). The argument *mention* may have one of the following values:

*t* (default value) — The name of the topic is to be mentioned;

*nil* — It is not necessary to mention the topic's name as far as the Tutoring Strategist is concerned. This directive may be overridden by considerations applied by the Comprehension-Processes Module; or



**Fig. 2app.1: Input Sequence of Technical Utterances**

The anticipated number of equations — Since a tutor is seldom certain of the exact number of equations he wishes to present, and usually has a range in mind (e.g., few equations, several, many, an entire session), this argument shall be a pointer to a particular range in a range-list. For example, given the following range-list:

1	2	3	4	5	6
0-1	2-4	5-8	9-12	13-17	18-30

If mention=3, it means that the tutor plans to present between 5 and 8 exercises in a given topic.

((**equation** *equation-representation*) (*existing-alternatives* *mentioned-alternatives* *certainty*) (*similar-equations* *qualifier*) *complexity*).

This utterance presents the current equation. The *equation-representation* argument appears in string form (e.g., " $x^2-3x-4=0$ "). The *existing-alternatives* argument contains the number of different approaches that can be used to solve this equation, while the *mentioned-alternatives* argument represents the number of alternatives the tutor is about to present. The *certainty* argument shall have value  $t$ , if all the solution alternatives to be presented are certain to solve the given equation; otherwise its value shall be *nil*. Notice that the certainty of a solution not only depends on the method being considered (e.g., a quadratic equation can always be solved by removing parentheses, collecting terms, and applying the quadratic formula), but also on the particular equation and the knowledge status of the student. For example, if a particular solution alternative was not guaranteed to solve an equation, but eventually did, then if the next equation has the same pattern, this sequence of operations is expected to solve it.

The *similar-equations* argument may have one of the following values:

A **Pattern-name** — This is the name of a well-known pattern, like a linear canonic form (" $ax+b=c$ "), or a quadratic canonic form (" $ax^2+bx+c=0$ "), which the student should be able to recognize; or

A list of equation numbers — This list contains references to equations which were given lately, and the tutor recognizes as having a pattern similar to the current equation. As opposed to the previous value, in general, the referenced equations would not have been remembered if they had not been presented recently.

The *qualifier* contains information which restricts the similarity between the current equation and similar ones. For example, an unqualified similarity statement could be: “This equation is similar to the previous one.” While the same statement accompanied by a qualifier would be: “This equation is similar to the previous one, however, it has another term.” In the present implementation, the qualifier has a string form. It is well known, that this format is not conducive to the generation of fluent text. Nevertheless, its mere presence enables us to recognize some features which are needed for the generation of cogent text (see section 5.2).

Finally, the *complexity* argument is a number between 0 and 1, which measures the difficulty of the current equation (where 0 represents the easiest equation and 1, the most difficult).

((*alternative solution-certainty*) *alternative-number* (*actual-steps mentioned-steps*)).

This utterance signals the beginning of a solution to the equation. Like before, the *solution-certainty* argument has value *t*, if the tutor wishes to convey that the forthcoming sequence of operations is certain to solve a given equation. Otherwise, its value is *nil*.

Next, the *alternative-number* argument contains the sequential number of the current alternative. Finally, the *actual-steps* argument contains the total number of operations applied to solve the equation, while the *mentioned-steps* argument contains the number of steps that shall be mentioned.

((rule *rule-name rule-slots*) *rule-number number-of-applications similar-usage repeat approach*).

Throughout this work, we shall refer to high-level algebraic operations like removing parentheses, collecting terms, factoring out common factors, etc, as *rules*.

*rule-slots* contains a possibly empty list of arguments which are related to the application of the rule. The types of the arguments remind us of the slots used in Conceptual Dependencies (Schank and Riesbeck 1981), e.g., actor, action, object, from, to. However, their contents differ from traditional CD values, in that they are oriented towards the generation of text concerning mathematical entities. The following example contains a stylized representation of the technical part of a rule TU (for a detailed description of the rule-slots see Appendix 3).

(rule factor-out ((object "3x") (from "the first and second terms")))

This representation enables us to generate a sentence like: "We factor out 3x from the first and second term."

The *rule-number* argument contains the number of the current rule in the sequence of operations.

There are some rules which are usually applied only once for a given instance (e.g., quadratic formula), thus requiring a constant number of computations. Whereas the number of computations required for other rules (e.g., remove parentheses, collect terms) depends on the number of times they are applied. The *number-of-applications* argument contains the number of times the current rule is applied, thus enabling us to distinguish between inherently lengthy operations and computationally short operations, which due to repetitions demand many computations.

The *similar-usage* argument may have one of the following values:

A list of equation numbers containing the numbers of equations in which the current rule was applied under similar circumstances;

*t* — If a list of similar application instances was already given in a previous rule or a list of *similar-equations* was given in the equation TU, and the current rule was used under similar circumstances than the equations in these lists; or

*nil* — If the tutor does not recall using the current rule under similar circumstances in previous equations.

Notice that it is not sufficient for a rule merely to have been used previously in order to merit a non-null *similar-usage* argument. The manner in which it is applied, and the mathematical entities on which it is applied have to be similar to a previous pattern and mathematical entities, respectively.

The *repeat* argument resembles the *similar-usage* argument, however in this case, the similarity of use pertains to the same equation. It can either be *nil*, or it can contain a list of tuples such each tuple has the format (*alternative-number rule-number*).

When learning to solve algebraic equations, the main challenge of an equation typically appears at the beginning. After having successfully applied one or more rules, the equation reaches a format the student is already familiar with, and the solution process becomes routine. The last argument, namely *approach*, serves to distinguish between the first part in the solution, and the second one. It has value *t*, if the current rule is regarded as one of the rules used to attack the current equation, and *nil*, otherwise.

((**pattern** *pattern-slots*)).

This TU contains information about a successfully matched pattern which enables us to apply the rule under consideration. As stated above, it appears after a rule TU. The *pattern-slots* have the same format as the *rule-slots*, and are described in detail in Appendix 3. A stylized representation of the technical part of a rule TU and its accompanying pattern TU is given in the following example:

```
(rule factor-out ((object "x2")))
(pattern ((actor object) (object "the rest of the terms on the lhs")))
```

Notice that the actor slot of the pattern refers to the object of the rule. This representation enables us to generate a sentence containing pronouns, e.g., "Since  $x^2$  is a factor common to the rest of the terms on the left hand

side, we factor it out.’’

((**expectation** *applicable-case expectation-slots*)).

This TU contains information regarding the expectations awakened when applying a rule.

The *applicable-case* argument selects the appropriate expectation from a list of expectations which accompany each rule in the database. The *expectation-slots* argument is similar to the *rule-slots* argument and is described in detail in Appendix 3. The following example contains a stylized representation of the technical part of an expectation TU:

(**expectation** **common-factor** ((*object* ‘‘other terms’’)))

Through this representation we can generate a clause such as ‘‘hoping to get a factor common to other terms.’’

((**result** *equation-representation*) *similar-result* (*similar-equations* *qualifier*) *expectation-fulfilled*).

This TU contains an equation representing the result of an algebraic operation. Like for equation TUs, the *equation-representation* has a string format.

The values accepted by the *similar-result* argument are the same as the values accepted by the *similar-usage* rule-argument, namely a list of equation numbers, *t* or *nil*. If this argument has a non-null value, it indicates that a similar result was encountered in a previous equation. Moreover, a value of *t* signals that the current result is in keeping with a similarity pointed out by a previous argument. Through this argument, the *similar-usage* rule argument and



the *similar-equations* equation argument, a tutor can express the fact that a solution to an equation is similar to a previous solution and also call attention to an eventual divergence between the solution paths.

The values accepted by the (*similar-equations qualifier*) arguments are the same as the values accepted by these arguments in equation TUs. A non-null argument of this type means that the current result is similar to a previously encountered equation. This argument differs from the *similar-result* argument in that it references an equation, as opposed to a result that was obtained during the solution of an equation. It serves to indicate that the approach to be used after obtaining the result under consideration should be the same as the approach used for solving the equations in the argument.

Finally, the *expectation-fulfilled* argument may have one of the following values: *t*, if a previously stated expectation was fulfilled and *nil*, otherwise. If this argument is omitted, it either means that no expectations were previously stated, or that the current result is irrelevant to any previously stated expectations, i.e., neither a violation or a realization of a stated expectation has been evidenced.

((**finish** *solved*)).

This TU signals the completion of a solution path. The argument *solved* was value *t* or is omitted, if the equation was successfully solved, and *nil*, otherwise.

((**continue**) *list-of-remaining-rules remaining-length mention-rules*).

This TU signals the eventual successful completion of a solution alternative,

i.e., the last steps may be omitted, and it should be clear to the student how to solve the equation from this point on. The *list-of-remaining-rules* argument contains a list of the operations to be performed in order to complete the solution of the equation. If an element in this list is also a list, its elements are alternative operations. For example, the following *list-of-remaining-rules* represents sequence of operations: “remove parentheses, collect terms, and then apply the quadratic formula or complete the square.”

(remove-parentheses collect-terms

(quadratic-formula completion-to-square))

As stated before, each high-level algebraic operation has a length attribute, which consists of the number of computations associated with it. The *remaining-length* argument contains the sum of the lengths of all the remaining operations, taking into consideration the number of times they have been applied.

Finally, the *mention-rules* argument has value *t*, if the *list-of-remaining-rules* should be mentioned, and *nil*, otherwise.

((method) *general-state specific-state* <sup>↑</sup>*shift-context*).

This TU indicates that the tutor wishes to inform the student about the methods available to solve different types of equations. For a given topic or distinguished equation pattern, the Tutoring Strategist may transmit information about general methods, specific methods or both.

The argument *general-state* has value *nil*, if the tutor does not wish to make any statement regarding general solution methods, or (*general state*),

otherwise. Similarly, the argument *specific-state* has value *nil*, if no information is to be transmitted regarding particular methods of solution, and value (*specific state*), otherwise. Notice, however, that a method TU requires at least one of these arguments to be non-null.

The types of the method statements to be issued depend on the values of their *state* arguments. If *state* has value *t*, the names of the solution methods are to be explicitly stated. A null value, however, elicits a statement about the existence of solution methods or lack thereof.

The *shift-context* argument may have one of the following values:

*(topic-name nil nil nil)* — Signals that the methods pertaining to a particular topic (e.g., linear, quadratic, third degree) are to be presented;

*t* — Signals that the methods pertaining to the topic under consideration are to be stated; or

*nil* (default) — Indicates that the methods to be stated pertain to the latest mathematical entity mentioned for which such methods are relevant. This entity can either be an equation with a distinguished pattern or a topic.

The following examples illustrate the semantics of the method TU:

*((method) (general nil) nil)* = “Give information regarding the existence of general methods.”

*((method) (general t) (specific nil))* = “Give a list of general methods, and

information regarding the existence of specific methods.”

## APPENDIX 3

### Arguments of the Technical Part of a Technical Utterance

The technical parts corresponding to a rule, pattern and expectation technical utterance have many elements in common. In the following sections we shall discuss their arguments in detail.

#### 3.1 Arguments of the Technical Part of a Rule Technical Utterance

When an algebraic operation is applied, the following types of arguments may be present:

**Object** — The object to which the rule is applied. For example, in “factor out  $x-1$ ” the object is “ $x-1$ .”

**To** — The target location of an algebraic operation, e.g., “transfer  $x$  to the left and side of the equation.”

**From** — The source location of an algebraic operation, e.g., “transfer terms from the right hand side.”

**Instrument** — The manner in which an algebraic operation is performed, e.g., “divide both sides of the equation by 3.”

**Target** — A certain form resulting from an algebraic operation, e.g., “rewrite  $-x-1$  as  $-(x+1)$ .”

The mathematical rule base supplies the arguments required and allowed by each rule, the order in which these arguments are to appear in the generated text, and also default values for the required arguments. For instance, if the rule *factor-out* was given with no arguments, the database supplies the object, yielding “factor out a common factor.”

Notice that the CD-slots *actor* and *action* are absent from the above given list. The reason for this is that the actor of an algebraic operation is the person performing it, and the action is the name of the operation itself.

Most of these arguments can be applied to mathematical entities on the left hand side and on the right hand side of an equation. Therefore they have the following format (*argument-type lhs rhs*).

Where either *lhs* or *rhs* could be *nil* (but not both of them at the same time). The format of *lhs* and *rhs* is discussed in Appendix 4.

### 3.2 Arguments of the Technical Part of a Pattern Technical Utterance

When referencing the pattern of a rule, the following slots are required:

**Actor** — This slot has to be supplied by the Tutoring Strategist, and can not have a default value. For example, the actor of a pattern could be: “ $x^2$ ” in the expression “ $x^2$  is a factor common to ... .”

**Object** — If this slot is absent, a default value can be retrieved from the rule base.

These slots differ from the ones used in the rule TU in two aspects. Firstly, together with an *action* slot, they are necessary and sufficient to create a legal pattern

TU. In addition, they accept two types of arguments:

*(slot-type lhs rhs)* , or

*(slot-type rule-slot-name)*

The last argument type is a reference to a slot in *rule-slots*. It is used to preclude duplication of reference, and enables the system to perform appropriate pronominalization.

### 3.3 Arguments of the Technical Part of an Expectation Technical Utterance

The format of the arguments required by *expectation-slots* is the same as the format of the *rule-slots* arguments. However, typically, *expectation-slots* require less arguments than *rule-slots*. Most expectation TUs need only an object argument, which can also be provided by the default value in the database, in addition to the action-slot, which is inherent in the expectation under consideration. Furthermore, there are instances of expectations which are independent of parameters, and therefore require no arguments. An instance of this case is illustrated by one of the purposes for removing parentheses, which is "to get simple terms only."

## APPENDIX 4

### Referencing Mathematical Entities

When referencing mathematical entities, we may either use an implicit representation (e.g., “the first and second terms”), an explicit representation (which contains copies of the entities referenced), or both. Therefore *lhs* and *rhs* may have one of the following formats:

*t* or *nil* or

*(implicit-representation explicit-representation)*

The value *t* indicates that an entire side of an equation is being referenced, the explicit representation contains a list of the elements we are referring to, and the implicit representation has the following format:

*(elements total-number-of-elements element-name mention-side other-information)*

Where:

*elements* contains the number of elements referenced by an algebraic operation, or an actual list of the positions of the elements referenced;

*total-number-of-elements* contains the total number of elements of the same type as the elements in the first argument;

*element-name* contains the name of the element. This could either be a simple name



like “term” or “factor” or a composite name, such as “common factor”;

*mention-side* has value *t* if the corresponding side should be mentioned, and *nil*, otherwise; and

*other-information* contains keywords which help refer to the mathematical entities under consideration. For example, “each,” “other,” “rest,” etc. This parameter is optional.

The following examples clarify the use of the implicit representation:

(2 3 term t) → “terms on the left/right hand side of the equation”

((1 3) 3 factor nil) → “the first and third factors”

This representation is geared towards the expression of simple mathematical entities, and does not allow us to refer to more complicated ones, like “the third factor in the first term.” Nevertheless, it is sufficient for most explanations required in the realm of high-school algebra, and enables us to illustrate the capabilities of the system. A more accurate representation could be easily substituted for the present one.

## APPENDIX 5

### Output of the Comprehension-Processes Module

The Comprehension-Processes Module adds requirement-codes corresponding to the MTUs to be generated for each technical utterance, and may add information to the technical part of a technical utterance. As mentioned in Chapter 4, it may also insert an alternatives technical utterance or delete an alternative technical utterance. In the following sections we present the modified technical part and the different MTU-codes, which may be produced by the Comprehension-Processes Module.

Before we proceed with the explanation, we would like to introduce the following notation:

The parameters in italics have to be substituted with arguments. For instance, a number between 0 and 1 has to appear in the position of the parameter *degree*.

A parameter enclosed in square brackets is optional.

If a list of parameters is separated by slashes, one of them has to be selected.

#### 5.1 Topic Extended-Message

##### Technical-part

(topic *topic-name*)

## MTU-codes

### Motivation

knowledge-preservation  
(increment-knowledge *degree*)  
(practice-reassure *number-of-previous-motivations*)  
(highlight-attributes *topic-name (attribute<sub>1</sub> attribute<sub>2</sub> ... attribute<sub>n</sub>)*)  
obligate

### Focus

(close *open-topic*)

### Implementation-mode

continue  
return  
new  
(known (*eqn<sub>1</sub> eqn<sub>2</sub> ... eqn<sub>k</sub>*))

### Length

add  
(premature-end *previous-topic*)  
(mention *range-of-equations*)  
apologize-length

### Complexity

(complex/simple *relative-complexity [(like-before (*eqn<sub>1</sub> eqn<sub>2</sub> ... eqn<sub>k</sub>*))]*)  
complexity-reassure

### Digression

close

## 5.2 Equation Extended-Message

### Technical-part

(equation *equation-representation* [t/nil])

### MTU-codes

#### Motivation

knowledge-preservation  
(increment-knowledge *degree*)  
(highlight-attributes *pattern-name (attribute<sub>1</sub> attribute<sub>2</sub> ... attribute<sub>n</sub>)*)  
obligate

**Focus**

(close *open-equation*)

open

**Implementation-mode**

new

(similar ((*eqn<sub>1</sub>* *eqn<sub>2</sub>* ... *eqn<sub>k</sub>*) [*qualifier*]))

(similar (*pattern-name* [*qualifier*]))

**Length**

apologize-length

(mention/exist *number-of-alternatives* t/nil)

**Complexity**

(complex/simple *relative-complexity* [(*like-before* (*eqn<sub>1</sub>* *eqn<sub>2</sub>* ... *eqn<sub>k</sub>*))])

complexity-reassure

**Digression**

close

**5.3 Alternatives Extended-Message****Technical Part**

(alternatives)

**MTU-codes****Motivation**

(*motivation-type* *number-of-rules*)

(*motivation-type*

[(*highlighted-attributes* *rule-name*  
(*attribute<sub>1</sub>* *attribute<sub>2</sub>* ... *attribute<sub>n</sub>*))])

where *motivation-type* ∈ {*knowledge-preservation*,  
(*increment-knowledge* *degree*)}

(*introduce-method*

(*highlight-attributes* *rule-name* (*attribute<sub>1</sub>* *attribute<sub>2</sub>* ... *attribute<sub>n</sub>*))  
[(*comparison-purposes* *number-of-compared-alternatives*)]])

**5.3 Alternative Extended-Message****Technical Part**

(alternative t/nil)

## MTU-codes

### Motivation

(*motivation-type number-of-rules*  
[(*another number-of-previous-alternatives-with-same-motivation*)]  
[*rule-name (attribute<sub>1</sub> attribute<sub>2</sub> ... attribute<sub>n</sub>)*])  
where *motivation-type* ∈ {knowledge-preservation,  
(increment-knowledge *degree*), attempted}  
(highlight-attributes *rule-name (attribute<sub>1</sub> attribute<sub>2</sub> ... attribute<sub>n</sub>)*)  
(obligate)

### Focus

(close *open-alternative*)  
open

### Length

apologize-length  
(mention/exist *number-of-operations [mentioned-operations]*)

### Temporal

(*alternative-number ordinal/cardinal*)  
last

### Affect

positive

### Digression

close

### Expectation-solution

(violation expectation t/nil)

## 5.4 Rule Extended-Message

### Technical Part

(rule *rule-name rule-slots*)

### MTU-codes

#### Motivation

knowledge-preservation  
(increment-knowledge *degree*)  
obligate  
(highlight-attributes *rule-name (attribute<sub>1</sub> attribute<sub>2</sub> ... attribute<sub>n</sub>)*)

**Focus**

(close *open-rule*)  
refer-motivation

**Implementation**

(again (*operation<sub>1</sub> operation<sub>2</sub> ... operation<sub>n</sub>*))  
(like-before alternative ((*alternative<sub>1</sub> operation<sub>1</sub>*) (*alternative<sub>2</sub> operation<sub>2</sub>*) ...  
(*alternative<sub>n</sub> operation<sub>n</sub>*)))  
(like-before equation (*equation<sub>1</sub> equation<sub>2</sub> ... equation<sub>n</sub>*))  
(known (*equation<sub>1</sub> equation<sub>2</sub> ... equation<sub>n</sub>*))  
new

**Length**

apologize-length  
(long-rule *rule-length* [(like-before (*equation<sub>1</sub> equation<sub>2</sub> ... equation<sub>n</sub>*))])  
(long-situation *situation-length*)

**Complexity**

(difficult/easy *relative-complexity*  
[(like-before (*equation<sub>1</sub> equation<sub>2</sub> ... equation<sub>n</sub>*))])  
complexity-reassure

**Temporal**

*rule-number*  
last

**Digression**

close

**Expectation-solution**

(violation expectation t/nil)

**Expectation-event**

(violation event [rule])

**Expectation-similarity**

(violation/realization (similarity (*alternative<sub>1</sub> alternative<sub>2</sub> ... alternative<sub>n</sub>*)))  
(violation dissimilarity)

**5.5 Pattern Extended-Message****Technical Part**

(pattern *pattern-slots*)

MTU-codes

**Implementation**

(like-before (*equation<sub>1</sub>* *equation<sub>2</sub>* ... *equation<sub>n</sub>*))

**5.6 Expectation Extended-Message**

Technical Part

(*expectation applicable-case expectation-slots*)

MTU-codes

**Implementation**

(like-before (*equation<sub>1</sub>* *equation<sub>2</sub>* ... *equation<sub>n</sub>*))

**5.7 Result Extended-Message**

Technical Part

(*result representation*)

MTU-codes

**Focus**

close

**Affect**

negative

**Expectation-result**

(*violation/realization expectation t/nil*)

**Expectation-similarity**

(*violation/realization (similarity (alternative<sub>1</sub> alternative<sub>2</sub> ... alternative<sub>n</sub>))*)

(*violation dissimilarity*)

## 5.8 Finish Extended-Message

### Technical Part

(finish t/nil)

### MTU-codes

**Focus**  
close

**Digression**  
close

**Affect**  
negative

## 5.9 Continue Extended-Message

### Technical Part

(continue [(*rule*<sub>1</sub> *rule*<sub>2</sub> ... *rule*<sub>*n*</sub>)])

### MTU-codes

**Focus**  
close

**Digression**  
close

## 5.10 Method Extended-Message

### Technical Part

(method *tu-type tu-name* t/nil [(*method*<sub>1</sub> *method*<sub>2</sub> ... *method*<sub>*n*</sub>)]  
(method *tu-type tu-name* t/nil not-teach/not-exist/nil)  
where *tu-type* ∈ {topic, equation}

### MTU-codes

**Implementation**  
(like-before (*equation*<sub>1</sub> *equation*<sub>2</sub> ... *equation*<sub>*n*</sub>))



also

**Category**

general

specific

**Digression**

(open *context*)

(change *context*)

close

**Affective**

negative

positive

**Expectation**

violation

**5.11 Report Extended-Message**

**Technical Part**

(report t/nil ((length *value*<sub>1</sub>) (complexity *value*<sub>2</sub>) (number-of-steps *value*<sub>3</sub>)  
(error-proneness *value*<sub>4</sub>)) (*rest-of-alternatives successful-alternatives*))

## APPENDIX 6

### Thresholds for the Generation of Estimational MTUs

In the following sections we shall discuss the thresholds used by FIGMENT to determine whether a student will benefit from receiving an estimational MTU. These thresholds were empirically reached, by comparing the result of using them with decisions reached by introspection.

#### 6.1 Thresholds for Complexity-Related MTUs

In order to determine the need for generating a complexity-related MTU, the following expression is calculated:

RELATIVE-COMPLEXITY — TALENT

Where:

$$RELATIVE-COMPLEXITY = \frac{COMPLEXITY-OF-TECHNICAL-UTTERANCE}{\sqrt{RELATIVE-EXPERTISE + 0.25}}$$

It is then compared with the following thresholds:

easy	difficult	very difficult
-0.4	0.1	0.3

The meaning of these thresholds is that if the relative complexity of a technical utterance exceeds a student's talent by more than 0.1, then FIGMENT anticipates

that the student shall have difficulty in processing the technical utterance under consideration, if he uses default computational power. If the relative complexity exceeds the student's talent by more than 0.3, then the student is very likely to fail in processing the technical utterance, even if he applies high computational power. Finally, if the relative complexity falls below the student's talent by more than 0.4, then the system predicts that the technical utterance shall be easily processed by the student.

Notice, however, that the above given formula is applied only if the student was already exposed to the utterance under consideration. Otherwise, the concept of RELATIVE-EXPERTISE is meaningless. In this case, only the complexity of the technical utterance and the talent of the student are taken into consideration.

Fig. 6app.1 contains some values of the complexity of a technical utterance, which would entail the generation of complexity-related MTUs for a student with a given talent and relative-expertise.

According to this table, if a student has a talent of 1, and a relative expertise of 1 in a given type of technical utterance, then if the complexity of this utterance is less than 0.67, he shall be advised of its simplicity. Notice that a student with talent 1 and expertise 1 in a particular technical utterance shall not require a complexity-related MTU advertising the difficulty of this utterance. A student whose talent is evaluated at 0.75, and has succeeded in processing half of the technical utterances of the type being considered, shall be advised of the difficulty of an utterance, if its complexity exceeds 0.74. The starred entries in the table contain measures of complexity for which no MTUs shall be generated despite the result of the above presented formula. As stated in section 4.4.1, the difficulty of technical utterances whose complexity falls below 0.2 shall not be advertised, regardless of the lack of talent or expertise of a

Talent	Relative Expertise	Complexity		
		Easy	Difficult	Very Difficult
1	1	0.67	-	-
1	0.75	0.6	-	-
1	0.5	0.51	0.96	-
1	0	0.3	0.55	0.65
0.75	1	0.39	0.96	-
0.75	0.75	0.35	0.85	-
0.75	0.5	0.3	0.74	0.91
0.75	0	0.17	0.43	0.53
0.5	1	0.11	0.67	0.89
0.5	0.75	0.1	0.6	0.8
0.5	0.5	0.08	0.52	0.7
0.5	0	0.05	0.3	0.4
0	1	-	0.12*	0.34*
0	0.75	-	0.1*	0.3*
0	0.5	-	0.09*	0.26*
0	0	-	0.05*	0.15*

Fig. 6app.1: Selected Values of Talent, Relative-Expertise and Complexity

student. Neither will FIGMENT produce a complexity-reassure MTU for technical utterances whose complexity is lower than 0.4.

## 6.2 Thresholds for Length-Related MTUs

FIGMENT decides whether to generate a length-related MTU, by comparing the RELATIVE-LENGTH of a technical utterance with a threshold, where the RELATIVE-LENGTH is calculated by the following formula:

$$RELATIVE-LENGTH = \frac{UTTERANCE-LENGTH}{\sqrt{\max(DILIGENCE, 0.2)}}$$

The threshold used to determine the need for a length-related MTU is defined as the greatest utterance length, which can be processed by a student with diligence 1, without causing him to experience negative affects. Unlike the thresholds used for generating complexity-related MTUs, each type of technical utterance has its own threshold. This is due to the fact that, the length of most technical utterances can be measured, and different technical utterances require different measurement units. The table 6app.2 contains the thresholds used to generate length-related MTUs.

utterance type	too lengthy	extremely lengthy	measurement units
topic	5	9	# of exercises
equation	3	5	# of alternatives
alternative	8	12	# of steps
rule	0.8	0.9	# of operations (normalized)
statement	0.8	0.9	# of clauses (normalized)

**Fig. 6app.2: Thresholds Used to Generate Length-related MTUs**

According to this table and the above given formula, if a student has a diligence of 1, then a length-related MTU would be issued for an equation with more than 3 alternatives. Table 6app.3 contains some values of the length of a technical utterance, which would entail the generation of a length-related MTU for a student with a given diligence.

According to the table in Fig. 6app.3, if a student has a diligence of 0.3, he will be notified of the length of a topic if the tutor is about to examine more than three

diligence	topic		equation		alternative		rule	
	too long	very long	too long	ver long	too long	very long	too long	very long
0.1*	2	3	1	2	3	4	0.26	0.29
0.2	3	5	2	3	4	6	0.36	0.41
0.3	3	5	2	3	5	7	0.44	0.5
0.4	4	6	2	4	6	8	0.51	0.57
0.5	4	7	3	4	6	9	0.57	0.64
0.6	4	7	3	4	7	10	0.62	0.7
0.7	5	8	3	5	7	11	0.67	0.76
0.8	5	9	3	5	8	11	0.72	0.81
0.9	5	9	3	5	8	12	0.76	0.86
1.0	6	10	4	6	9	13	0.81	0.91

**Fig. 6app.3: Some Values of Diligence and Length**

equations. If more than five equations are to be discussed, the tutor also has to extend his apologies. The system does not take into consideration diligence measures below 0.2, in order to avoid generating length announcements and apologies for short technical utterances (see starred entry in Fig. 6app.3).

## APPENDIX 7

### Rules Applied for Selecting a Motivation

When selecting a motivation FIGMENT's tries to imply the least possible lack of knowledge. According to this policy it will generally favour a motivation which highlights the attributes of a technical utterance over a knowledge-status related motivation. Among the latter, it will prefer an equation motivation over a topic motivation, and an upwardly-propagated method motivation over an equation motivation.

The following rules are applied to the structure produced by the motivation-determination process to determine the motivation for the entire equation. The Comprehension-Processes Module then checks if any alternative is not covered by the inheritance property and requires a separate motivation.

- i. If the student should be motivated to attend to the current topic, and more than one equation shall be mentioned, then motivation through the topic is mandatory.
- ii. Otherwise, if all the methods in the typical sequence of each alternative have been recently motivated, then all alternatives are considered to be motivated, and so is the entire equation. Notice, however, that this rule affects only the global motivation of the entire equation, and individual alternatives which are presented after the equation is solved still have to be motivated.

- iii. If a highlight-attributes topic motivation is to be presented, then a subset of the topic's attributes is selected from the Problem Solving Expert's domain knowledge. This selection process is based on the value of each attribute, i.e., the higher the value of an attribute, the better its chances to be chosen.
- iv. In order to imply the least possible lack of knowledge, the system prefers to motivate a student through a narrow area of knowledge, instead of a broader domain. This preference is expressed by the policy that only in 25% of the cases in which the topic motivation is not mandatory, a knowledge-status related motivation shall be attempted. In addition, no social motivation of the topic shall be performed at this point. The system shall resort to this type of motivation only if no other type is applicable.
- v. If an equation requires a highlight-attributes motivation, it shall be motivated in this manner if a highlight-attributes topic motivation was not presented, and in 60% of the rest of the cases. This motivation may seem redundant if a student is already motivated to attend to the topic, however, since a new type of equation is being introduced, a tutor would generally wish to inform the student of its attributes. Like for a topic motivation, the attributes of a distinguished equation pattern are selected from the Problem Solving Expert's domain knowledge.
- vi. If a topic motivation was selected, any other type of motivation required by the equation shall be ignored. Otherwise, a knowledge-status related motivation of the equation shall be attempted in 50% of the cases.
- vii. If neither a motivation through the topic nor through the equation was per-



formed, then an upwardly propagated motivation through the alternatives is tried. If successful, this type of motivation is incorporated in an alternatives extended message. This message contains only a motivational MTU-code, and is inserted in the extended file after the equation extended message. It is used to generate a sentence such as “This equation enables us to exercise several techniques which demand some more practice.”

- viii. Finally, if none of the attempted motivations succeeded, then the system tries again a knowledge-status related motivation of the equation, and if this fails, this type of motivation is attempted again for the topic. If the system does not meet with success once more, then in 60% of the cases the equation is motivated by obligation and in the rest of the cases, the topic is motivated by obligation.

When attempting to motivate a student to attend to an equation by means of an upwardly propagated motivation of its alternatives, the system applies the following rules.

- i. If one of the alternatives introduces a new technique by means of a highlight-attributes motivation, an *introduce method* motivation may be stated with the equation. This type of motivation creates an expectation for the immediate presentation of the new method. However, as stated in section 3.3, the Tutoring Strategist would generally postpone the introduction of a new technique to the last alternative. In this case, a violation of expectation MTU has to be generated, and the presence of the other alternatives has to be justified by means of a *comparison purposes* statement. This policy would yield a motivation statement like the following: “This equation enables us to introduce the very

important and useful technique of substitution, **but first** let us consider two other ways of solving this equation, **for comparison purposes.**"

- ii. Otherwise, if only an obligation motivation can be used for all the methods in a typical sequence of a solution alternative, then the upwardly propagated motivation is abandoned, since a social motivation could be used for the equation or the topic.
- iii. If these conditions are not satisfied, the system tries to motivate a student to attend to the equation by means of a knowledge-status related motivation applicable to at least one rule in the typical sequence of each alternative. If both types of knowledge-status related motivation are applicable to all alternatives, the type which appears more times is selected. For example, given the following alternatives and motivation MTU-codes for the rules in their typical sequences:

ALTERNATIVE<sub>1</sub> (method<sub>11</sub> *increment-knowledge*)  
 ALTERNATIVE<sub>2</sub> { (method<sub>21</sub> *increment-knowledge*),  
                   (method<sub>22</sub> *knowledge-preservation*)  
 ALTERNATIVE<sub>3</sub> { (method<sub>31</sub> *knowledge-preservation*),  
                   (method<sub>32</sub> *increment-knowledge*)

The increment-knowledge motivation would be selected, yielding a sentence like the following: "Through this equation we are able to exercise a few techniques which demand some more practice."

If there is only one solution alternative, with one rule in its typical sequence, a knowledge-status related motivation may be accompanied by some of this rule's attributes.

If FIGMENT decides to motivate a technical utterance by means of a knowledge-status related motivation, it applies the following rules to determine the need for accompanying MTUs.

- i. If an increment-knowledge motivation is required, and the technical utterance under consideration was previously motivated by means of an increment-knowledge motivation, then a consolatory *practice-reassure* MTU has to be generated, since the student has already encountered this technical utterance numerous times. This rule enables the system to produce a statement like the following: "I know you have exercised this type of equation many times, but it still demands some more practice."
- ii. A knowledge-status related motivation may be accompanied by some attributes of the technical utterance under consideration. These attributes are selected in the manner described above if they have never been stated before, or in 30% of the rest of the cases.

Finally, after the motivation MTU-code for the equation has been selected, each alternative which is not automatically motivated through the inheritance relation, has to be separately motivated. The following rules perform this task.

- i. If a student was motivated to attend to the equation by means of an upwardly propagated motivation of its alternatives, then, if an *introduce method* motivation was used, the commentary should be completed by a statement comparing the performance measures of the alternatives. Otherwise, if an upwardly propagated knowledge-status related motivation of the applied rules was used, the motivation has to be repeated for all the alternatives but the first one, yielding

motivational statements like the following: "Another alternative through which we can practice a couple of techniques we have not seen for a while consists of the following operations."

- ii. If the equation was directly motivated through its topic or its pattern, and if an alternative contains a new method which is motivated by means of a highlight-attributes motivation, then the system selects between one of the following ways to introduce this method (recall that if the equation has not been solved yet, a motivation for an alternative is not necessary, and the rule shall be accompanied by its attributes).

In most cases, the attributes of this method will be stated while introducing the alternative. For example, "Through the following alternative we shall introduce a very interesting technique, namely Patt's guessing method for solving quadratic equations." Next, if this technique is not the first one to be applied, an expectation violation MTU has to precede the first rule in the current alternative, and the method being motivated has to be accompanied by a focal MTU, which indicates that this is the method being referred to in the motivational statement. This rhetorical rule would produce the MTU-codes needed to generate the following sentences: "But first we remove parentheses from the first and second terms on the left hand side. Next we collect terms. Finally, we apply Patt's guessing method for solving quadratic equations. **This is the method I was talking about before.**"

In the rest of the cases, the alternative is introduced with an obligation motivation, by means of a sentence like the following: "Another solution I

would like you to try consists of the following steps,” and the highlighted attributes are presented together with the technique being motivated.

In any event, if the alternative solves the equation, the commentary is completed by a comparison between the performance measures of the alternatives.

- iii. Finally, if a new technique is not being introduced, the student was motivated to attend to the equation through its topic or pattern, and the equation is already solved, then a knowledge-status related or social motivation has to be presented for each remaining solution alternative. The selected motivation corresponds to the first method in the typical sequence of the alternative under consideration. Any other techniques in the typical sequence which share this motivation are counted, in order to enable the system to express the motivation in the proper grammatical person. This rule might yield a sentence like the following: “The second alternative enables us to exercise a few rules which demand plenty of practice.” If a knowledge-status related motivation is not applicable, an *attempted* motivation MTU-code is generated, to signal that the method being motivated is already known to the student. In this manner, the system is able to generate a motivational sentence such as “Another alternative you might have thought of consists of the following steps.”

If there is only one technique in the typical sequence of the current alternative, some of its attributes may be stated together with a knowledge-status dependent or social motivation.

## APPENDIX 8

### Typical Output of the Comprehension-Processes Module and Sentence Composer

The following sections contain typical examples of actual listings of the input to FIGMENT, the output of the Comprehension-Processes Module, and the output of the Sentence Composer. The notation “^” is used by FIGMENT for exponentiation, e.g.,  $x^2 = x^2$ .

#### 8.1 A Linear Equation

##### Input to FIGMENT

```
1 - ((topic linear))
2 - ((equation "3(x-1)-3=12") (2 2 t) nil 0.3)
3 - ((alternative t) 1 (4 4))
4 - ((rule remove_parentheses nil) 1 2 nil nil t)
5 - ((rule transfer_term nil) 2 2 nil nil nil)
6 - ((rule collect_terms nil) 3 3 nil nil nil)
7 - ((rule constant_divide nil) 4 2 nil nil nil)
8 - ((result "x=6") nil nil)
9 - ((finish))
10 - ((alternative t) 2 (4 4))
11 - ((rule constant_divide nil) 1 3 nil nil t)
12 - ((rule collect_terms nil) 2 1 nil ((1 3)) nil)
13 - ((result "x-2=12") nil nil)
14 - ((rule transfer_term nil) 3 1 nil nil nil)
15 - ((rule collect_terms nil) 4 1 nil ((2 2)) nil)
16 - ((finish))
```

##### Listing of the Output of the Comprehension-Processes Module

```
((topic linear))
activity: length (affect: ok)
activity: focus (affect: ok)
activity: focus (affect: confusion - no prepared space for topic)
activity: implementation (affect: disrespect - you put me in addition mode)
activity: motivation (affect: ok)
```

activity: temporary-focus (affect: ok)  
 ((topic linear) (open) ((known (1 2 3 4))) nil nil nil nil nil nil nil nil)  
 ((equation "3(x-1)-3=12") (2 2 t) nil 0.3)  
 activity: focus (affect: confusion - no prepared space for equation)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: motivation (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((equation | 3( x - 1 ) - 3 = 12|) (open) nil nil nil ((simple -0.5)) nil nil nil nil nil)  
 ((alternative t) 1 (4 4))  
 activity: length (affect: ok)  
 activity: focus (affect: ok)  
 ((rule remove\_parentheses nil) 1 2 nil nil t)  
 activity: focus (affect: ok)  
 activity: implementation (affect: disrespect - you put me in addition mode)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 activity: motivation (affect: ok)  
 ((rule remove\_parentheses nil) nil ((known (4))) nil nil ((easy -0.4422291236000336))  
 nil nil nil (1) nil)  
 ((rule transfer\_term nil) 2 2 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule transfer\_term nil) nil nil nil nil nil nil nil nil (2) nil)  
 ((rule collect\_terms nil) 3 3 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule collect\_terms nil) nil nil nil nil nil nil nil nil (3) nil)  
 ((rule constant\_divide nil) 4 2 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: disrespect - you put me in addition mode)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule constant\_divide nil) nil ((known (2 3 4 4))) nil nil  
 ((easy -0.6 (like before (2 3 4)))) nil nil nil (last) nil)  
 ((result "x=6") nil nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result | x = 6|) nil nil nil nil nil nil nil nil nil nil)  
 ((finish))

activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((finish) nil nil nil nil nil nil nil nil nil)  
 ((alternative t) 2 (4 4))  
 activity: length (affect: ok)  
 activity: focus (affect: confusion - previous alternative is still in focus)  
 activity: temporary-focus (affect: ok)  
 ((alternative t) ((close 1) open) nil nil nil nil nil nil nil ((2 cardinal)) nil)  
 ((rule constant\_divide nil) 1 3 nil nil t)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: boredom - too many operations)  
 activity: length (affect: extreme boredom)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 activity: motivation (affect: loss of attention)  
 ((rule constant\_divide nil) nil nil nil nil nil ((long\_situation 0.9) (apologize length))  
 nil nil (1) nil)  
 ((rule collect\_terms nil) 2 1 nil ((1 3)) nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: disrespect - you put me in addition mode)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule collect\_terms nil) nil ((like\_before alternative ((1 3)))) nil nil nil nil  
 nil nil (2) nil)  
 ((result "x-2=12") nil nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result | x - 2 = 12|) nil nil nil nil nil nil nil nil nil nil)  
 ((rule transfer\_term nil) 3 1 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule transfer\_term nil) nil nil nil nil nil nil nil nil (3) nil)  
 ((rule collect\_terms nil) 4 1 nil ((2 2)) nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: disrespect - you put me in addition mode)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule collect\_terms nil) nil ((again (2))) nil nil nil nil nil nil (last) nil)  
 ((finish))  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((finish) nil nil nil nil nil nil nil nil nil)



## Output File of the Comprehension-Processes Module

- 1 - ((topic linear) (open) ((known (1 2 3 4))) nil nil nil nil nil nil nil nil)
- 2 - ((equation | 3( x - 1 ) - 3 = 12|) (open) nil nil nil ((simple -0.5)) nil nil nil nil nil)
- 3 - ((rule remove\_parentheses nil) nil ((known (4))) nil nil ((easy -0.4422291236000336)) nil nil nil (1) nil)
- 4 - ((rule transfer\_term nil) nil nil nil nil nil nil nil nil (2) nil)
- 5 - ((rule collect\_terms nil) nil nil nil nil nil nil nil nil (3) nil)
- 6 - ((rule constant\_divide nil) nil ((known (2 3 4 4))) nil nil ((easy -0.6 (like\_before (2 3 4)))) nil nil nil (last) nil)
- 7 - ((result | x = 6|) nil nil nil nil nil nil nil nil nil nil)
- 8 - ((finish) nil nil nil nil nil nil nil nil nil nil)
- 9 - ((alternative t) ((close 1) open) nil nil ((obligate t) 1 nil (highlight\_attributes constant\_divide ((useful 0.8 (like\_before (2)))))) nil nil nil nil ((2 cardinal)) nil)
- 10 - ((rule constant\_divide nil) nil nil nil nil nil ((long\_situation 0.9) (apologize length)) nil nil (1) nil)
- 11 - ((rule collect\_terms nil) nil ((like\_before alternative ((1 3)))) nil nil nil nil nil nil (2) nil)
- 12 - ((result | x - 2 = 12|) nil nil nil nil nil nil nil nil nil nil)
- 13 - ((rule transfer\_term nil) nil nil nil nil nil nil nil nil (3) nil)
- 14 - ((rule collect\_terms nil) nil ((again (2))) nil nil nil nil nil nil (last) nil)
- 15 - ((finish) nil nil nil nil nil nil nil nil nil nil)

## Output of the Sentence Composer

Let us look at the subject of linear equations, which you have seen a couple of times. Here is a rather simple equation:

$$3(x - 1) - 3 = 12$$

First, you remove parentheses. You have seen this technique before, and it is easy. Next, you transfer terms and collect terms. Finally, you divide both sides by a constant. You have seen this method a few times, and, as was said a couple of times, it is very simple. Through this step you get the following result:

$$x = 6$$

Let us now examine another alternative. By means of it you can exercise the technique of division by a constant, which, as was mentioned in exercise number 2, is quite useful.

First, you divide both sides of the equation by a constant. This method requires plenty of computations in this case. Afterwards, you collect terms, like in the last alternative, which yields:

$$x - 2 = 12$$

You continue by transferring terms. Finally, you collect terms once again.

## 8.2 A Quadratic Equation

### Input to FIGMENT

```

1 - ((topic quadratic))
2 - ((equation "(x-3)^2 - 4(x-3) -12 = 0") (3 3 t) nil 0.7)
3 - ((alternative t) 1 (4 3))
4 - ((rule factor_out ((object (nil ((x-3)))) (from (((1 2) 3 term nil)))))) 1 2 nil nil t)
5 - ((pattern ((actor object) (object from))))
6 - ((expectation common_factor ((object ((1 3 term nil rest))))))
7 - ((result "(x-3)(x-7) -12=0") nil nil nil)
8 - ((rule remove_parentheses nil) 2 4 nil nil nil)
9 - ((rule collect_terms nil) 3 2 nil nil nil)
10 - ((result "x^2 -10x +9=0") nil nil)
11 - ((continue) (quadratic formula) 0.7 nil)
12 - ((alternative t) 2 (3 2))
13 - ((rule remove_parentheses nil) 1 5 nil nil t)
14 - ((rule collect_terms nil) 2 3 nil nil nil)
15 - ((result "x^2 -10x +9=0") nil nil)
16 - ((continue) (quadratic formula) 0.7 nil)
17 - ((alternative t) 3 (4 4))
18 - ((rule substitute_expression ((object (nil (y))) (for (nil ((x-3)))))) 1 2 nil nil t)
19 - ((pattern ((actor (nil (x))) (object (nil ((x-3))))))
20 - ((expectation canonic_expression))
21 - ((result "y^2-4y-12=0") nil nil)
22 - ((rule quadratic_formula nil) 2 1 nil nil nil)
23 - ((result "y=6") nil nil)
24 - ((result "y=-2") nil nil)
25 - ((rule substitute_back ((object (nil ((x-3)))))) 3 2 nil nil nil)
26 - ((result "x-3=6") nil nil)
27 - ((result "x-3=-2") nil nil)
28 - ((rule transfer_term ((object ((1 1 constant nil) (|-3|))) (to t))) 4 2 nil nil nil)
29 - ((result "x=9") nil nil)
30 - ((result "x=1") nil nil)
31 - ((finish))

```

### Listing of the Output of the Comprehension-Processes Module

```

((topic quadratic))
activity: length (affect: ok)
activity: focus (affect: ok)
activity: focus (affect: confusion - no prepared space for topic)
activity: implementation (affect: disrespect - you put me in addition mode)
activity: motivation (affect: ok)
activity: temporary-focus (affect: ok)
((topic quadratic) (open) (continue) nil nil nil nil nil nil nil)

```

((equation "(x-3)^2 - 4(x-3) - 12 = 0") (3 3 t) nil 0.7)  
 activity: focus (affect: confusion - no prepared space for equation)  
 activity: implementation (affect: ok)  
 activity: length (affect: boredom - too many alternatives)  
 activity: complexity (affect: ok)  
 activity: motivation (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((equation | ( x - 3 )^2 - 4( x - 3 ) - 12 = 0|) (open) nil nil nil nil ((exist 3))  
   nil nil nil nil)  
 ((alternative t) 1 (4 3))  
 activity: length (affect: ok)  
 activity: focus (affect: confusion - what are you talking about?)  
 activity: temporary-focus (affect: ok)  
 ((alternative t) (open) nil nil nil nil nil nil ((1 ordinal)) nil)  
 ((rule factor\_out ((object (nil ((x-3)))) (from (((1 2) 3 term nil)))))) 1 2 nil nil t)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: motivation (affect: ok)  
 ((rule factor\_out ((object (nil ((x-3)))) (from (((1 2) 3 term nil)))))) nil nil nil nil nil  
   nil nil nil (1) nil)  
 ((pattern ((actor object) (object from))))  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((pattern ((actor object) (object from))) nil nil nil nil nil nil nil nil nil nil)  
 ((expectation common\_factor ((object ((1 3 term nil rest))))))  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: disrespect - you put me in addition mode)  
 ((expectation common\_factor ((object ((1 3 term nil rest)))))) nil ((like\_before (6)))  
   nil nil nil nil nil nil nil nil nil)  
 ((result "(x-3)(x-7) - 12=0") nil nil nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result | ( x - 3 )( x - 7 ) - 12 = 0|) nil nil nil nil nil nil nil ((violation expectation))  
   nil nil)  
 ((rule remove\_parentheses nil) 2 4 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: disrespect - you put me in addition mode)  
 activity: length (affect: boredom - too many operations)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule remove\_parentheses nil) nil ((known (4))) nil nil ((easy -0.4422291236000336))  
   ((long\_situation 0.8)) nil nil (2) nil)  
 ((rule collect\_terms nil) 3 2 nil nil nil)  
 activity: focus (affect: ok)

activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule collect\_terms nil) nil nil nil nil nil nil nil nil (3) nil)  
 ((result "x^2 -10x +9=0") nil nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result | x^2 - 10x + 9 = 0|) nil nil nil nil nil nil nil nil nil nil)  
 ((continue) (quadratic\_formula) 0.7 nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((continue) nil nil nil nil nil nil nil nil nil nil)  
 ((alternative t) 2 (3 2))  
 activity: length (affect: ok)  
 activity: focus (affect: confusion - previous alternative is still in focus)  
 activity: temporary-focus (affect: ok)  
 ((alternative t) ((close 1) open) nil nil nil nil nil nil nil nil ((2 ordinal)) nil)  
 ((rule remove\_parentheses nil) 1 5 nil nil t)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: boredom - too many operations)  
 activity: length (affect: extreme boredom)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 activity: motivation (affect: loss of attention)  
 ((rule remove\_parentheses nil) nil nil nil nil nil ((long\_situation 1.0)  
 (apologize length)) nil nil (1) nil)  
 ((rule collect\_terms nil) 2 3 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule collect\_terms nil) nil nil nil nil nil nil nil nil (2) nil)  
 ((result "x^2-10x+9=0") nil nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result | x^2 - 10x + 9 = 0|) nil nil nil nil nil nil nil nil nil nil)  
 ((continue) (quadratic\_formula) 0.7 nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((continue) nil nil nil nil nil nil nil nil nil nil)  
 ((alternative t) 3 (4 4))  
 activity: length (affect: ok)  
 activity: focus (affect: confusion - previous alternative is still in focus)  
 activity: temporary-focus (affect: ok)  
 ((alternative t) ((close 2) open) nil nil nil nil nil nil nil nil (last) nil)

```

((rule substitute_expression ((object (nil (y))) (for (nil ((x-3)))))) 1 2 nil nil t)
activity: focus (affect: ok)
activity: implementation (affect: ok)
activity: length (affect: ok)
activity: complexity (affect: boredom - too easy)
activity: temporary-focus (affect: ok)
activity: motivation (affect: curiosity - why a new rule?)
((rule substitute_expression ((object (nil (y))) (for (nil ((x-3)))))) nil nil nil nil
  ((easy -0.5)) nil nil nil (1) nil)
((pattern ((actor (nil (x))) (object (nil ((x-3))))))
activity: focus (affect: ok)
activity: temporary-focus (affect: ok)
activity: implementation (affect: ok)
((pattern ((actor (nil (x))) (object (nil ((x-3)))))) nil nil nil nil nil nil nil nil nil)
((expectation canonic expression))
((result "y^2-4y-12=0") nil nil)
activity: focus (affect: ok)
activity: temporary-focus (affect: ok)
activity: implementation (affect: ok)
((result | y^2 - 4y - 12 = 0|) nil nil nil nil nil nil nil nil nil)
((rule quadratic_formula nil) 2 1 nil nil nil)
activity: focus (affect: ok)
activity: implementation (affect: ok)
activity: length (affect: ok)
activity: complexity (affect: ok)
activity: temporary-focus (affect: ok)
((rule quadratic_formula nil) nil nil nil nil nil nil nil nil (2) nil)
((result "y=6") nil nil)
activity: focus (affect: ok)
activity: temporary-focus (affect: ok)
activity: implementation (affect: ok)
((result | y = 6|) nil nil nil nil nil nil nil nil nil)
((result "y=-2") nil nil)
activity: focus (affect: ok)
activity: temporary-focus (affect: ok)
activity: implementation (affect: ok)
((result | y = - 2|) nil nil nil nil nil nil nil nil nil)
((rule substitute_back ((object (nil ((x-3)))))) 3 2 nil nil nil)
activity: focus (affect: ok)
activity: implementation (affect: ok)
activity: temporary-focus (affect: ok)
((rule substitute_back ((object (nil ((x-3)))))) nil nil nil nil nil nil nil nil (3) nil)
((result "x-3=6") nil nil)
activity: focus (affect: ok)
activity: temporary-focus (affect: ok)
activity: implementation (affect: ok)
((result | x - 3 = 6|) nil nil nil nil nil nil nil nil nil)
((result "x-3=-2") nil nil)
activity: focus (affect: ok)
activity: temporary-focus (affect: ok)

```

activity: implementation (affect: ok)  
 ((result |  $x - 3 = -2$ ) nil nil nil nil nil nil nil nil nil nil)  
 ((rule transfer\_term ((object ((1 1 constant nil) (-3))) (to t))) 4 2 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule transfer\_term ((object ((1 1 constant nil) (-3))) (to t))) nil nil nil nil nil nil nil  
 nil (last) nil)  
 ((result "x=9") nil nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result |  $x = 9$ ) nil nil nil nil nil nil nil nil nil nil)  
 ((result "x=1") nil nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result |  $x = 1$ ) nil nil nil nil nil nil nil nil nil nil)  
 ((finish))  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((finish) nil nil nil nil nil nil nil nil nil nil)

### Output File of the Comprehension-Processes Module

- 1 - ((topic quadratic) (open) (continue) nil nil nil nil nil nil nil nil)
- 2 - ((equation |  $(x - 3)^2 - 4(x - 3) - 12 = 0$ ) (open) nil nil nil nil ((exist 3))  
nil nil nil nil)
- 3 - ((alternative t) (open) nil nil nil nil nil nil nil ((1 ordinal)) nil)
- 4 - ((rule factor\_out ((object (nil ((x-3)))) (from (((1 2) 3 term nil)))))) nil nil nil nil nil  
nil nil nil (1) nil)
- 5 - ((pattern ((actor object) (object from))) nil nil nil nil nil nil nil nil nil nil)
- 6 - ((expectation common\_factor ((object ((1 3 term nil rest)))))) nil ((like\_before (6)))  
nil nil nil nil nil nil nil nil)
- 7 - ((result |  $(x - 3)(x - 7) - 12 = 0$ ) nil nil nil nil nil nil nil ((violation expectation))  
nil nil)
- 8 - ((rule remove\_parentheses nil) nil ((known (4))) nil nil  
((easy -0.4422291236000336)) ((long\_situation 0.8)) nil nil (2) nil)
- 9 - ((rule collect\_terms nil) nil nil nil nil nil nil nil (3) nil)
- 10 - ((result |  $x^2 - 10x + 9 = 0$ ) nil nil nil nil nil nil nil nil nil nil)
- 11 - ((continue) nil nil nil nil nil nil nil nil nil nil)
- 12 - ((alternative t) ((close 1) open) nil nil ((obligate t) 1 nil) nil nil nil nil ((2 ordinal))  
nil)
- 13 - ((rule remove\_parentheses nil) nil nil nil nil nil ((long\_situation 1.0))  
(apologize length)) nil nil (1) nil)
- 14 - ((rule collect\_terms nil) nil nil nil nil nil nil nil (2) nil)
- 15 - ((result |  $x^2 - 10x + 9 = 0$ ) nil nil nil nil nil nil nil nil nil nil)

- 16 - ((continue) nil nil nil nil nil nil nil nil nil nil)
- 17 - ((alternative t) ((close 2) open) nil nil  
 ((highlight\_attributes substitute\_expression ((efficient 0.8)))) nil nil nil nil  
 (last) nil)
- 18 - ((rule substitute\_expression ((object (nil (y))) (for (nil ((x-3)))))) nil nil nil nil  
 ((easy -0.5)) nil nil nil (1) nil)
- 19 - ((pattern ((actor (nil (x))) (object (nil ((x-3)))))) nil nil nil nil nil nil nil nil nil nil)
- 20 - ((result | y^2 - 4y - 12 = 0|) nil nil nil nil nil nil nil nil nil nil)
- 21 - ((rule quadratic\_formula nil) nil nil nil nil nil nil nil nil (2) nil)
- 22 - ((result | y = 6|) nil nil nil nil nil nil nil nil nil nil)
- 23 - ((result | y = - 2|) nil nil nil nil nil nil nil nil nil nil)
- 24 - ((rule substitute\_back ((object (nil ((x-3)))))) nil nil nil nil nil nil nil nil (3) nil)
- 25 - ((result | x - 3 = 6|) nil nil nil nil nil nil nil nil nil nil)
- 26 - ((result | x - 3 = - 2|) nil nil nil nil nil nil nil nil nil nil)
- 27 - ((rule transfer\_term ((object ((1 1 constant nil) (-3)))) (to t)) nil nil nil nil nil  
 nil nil nil (last) nil)
- 28 - ((result | x = 9|) nil nil nil nil nil nil nil nil nil nil)
- 29 - ((result | x = 1|) nil nil nil nil nil nil nil nil nil nil)
- 30 - ((finish) nil nil nil nil nil nil nil nil nil)
- 31 - ((report t ((length -1.7) (number\_of\_steps 1.5) (complexity -0.5)  
 (error\_proneness -0.55)) (2 2)) nil nil nil nil nil nil nil nil nil)

### Output of the Sentence Composer

We shall go on with the topic of quadratic equations. An equation follows:

$$(x - 3)^2 - 4(x - 3) - 12 = 0$$

There are three ways of solving this equation. The first alternative consists of the following operations:

First, since x-3 is a factor common to the first and second terms, we factor it out from these terms. As you know, we perform this step hoping to get a factor common to the remaining term. Contrary to what we were expecting, it arrives at:

$$(x - 3)(x - 7) - 12 = 0$$

Nevertheless, we can still eliminate parentheses. We saw this method in equation number 4, it is easy, but requires a lot of computations in this case. Thereafter, we collect terms, which yields the following result:

$$x^2 - 10x + 9 = 0$$

From here you can complete the solution of the equation by yourself.

Let us now examine the second way to solve this exercise. Through this alternative we can go over a solution, which you might have considered.

First, we get rid of parentheses. This technique demands plenty of calculations in this situation. Next, we collect terms, arriving at:

$$x^2 - 10x + 9 = 0$$

From this point you can obtain the solution by yourself.

We shall now consider the last alternative. This approach enables us to introduce the method of substitution, which is quite efficient. First, we substitute  $y$  for  $x-3$ , because  $x$  appears only in expression  $x-3$  and  $x-3$  appears more than once in the equation. This technique is rather simple. Through it we get the following result:

$$y^2 - 4y - 12 = 0$$

We go on by applying the quadratic formula, arriving at:

$$y = 6 \text{ or } y = -2$$

We continue by substituting back  $x-3$ , yielding the following result:

$$x - 3 = 6 \text{ or } x - 3 = -2$$

Finally, we transfer the constant, namely  $-3$ , to the left hand side of the equation, obtaining:

$$x = 9 \text{ or } x = 1$$

### 8.3 A Third Degree Equation

#### Input to FIGMENT

```

1 - ((topic third_degree))
2 - ((method (general t) (specific t)))
3 - ((equation "x^3-x^2-x+1=0") (3 1) nil 0.8)
4 - ((alternative nil) 1 (6 4))
5 - ((rule factor_out ((object (nil (x^2)))))) 1 2 nil nil t)
6 - ((pattern ((actor object) (object (((1 2) 4 term nil) nil))))))
7 - ((expectation common_factor ((object ((2 2 term nil rest) nil))))))
8 - ((result "x^2(x-1) - x+1=0") nil nil)
9 - ((rule rewrite_expression ((object (nil (-x+1))) (target (nil (|-(x-1)|)))))) 2 1 nil nil nil)
10 - ((result "x^2(x-1) -(x-1)=0") nil nil t)
11 - ((rule factor_out ((object (nil (x-1)))))) 3 2 nil nil nil)
12 - ((result "(x-1)(x^2-1)=0") nil nil)
13 - ((rule factorization_formula1 ((to (nil (x^2-1)))))) 4 1 nil nil nil)
14 - ((result "(x-1)^2(x+1)=0") nil nil)
15 - ((continue) (product_of_factors) 0.7 t)

```



## Listing of the Output of the Comprehension-Processes Module

```

((topic third_degree))
activity: length (affect: ok)
activity: focus (affect: ok)
activity: focus (affect: confusion - no prepared space for topic)
activity: implementation (affect: ok)
activity: motivation (affect: loss of attention - why a new topic?)
activity: temporary-focus (affect: ok)
((topic third_degree) (open) nil nil nil nil nil nil nil nil)
((method (general t) (specific t)))
activity: temporary-focus (affect: ok)
activity: implementation (affect: ok)
((method topic third_degree t not_teach) nil nil general nil nil nil nil nil nil negative)
activity: implementation (affect: ok)
((method topic third_degree nil ((factor_out) (factorization_formula))) nil nil specific
  nil nil nil ((violation expectation)) nil nil)
((equation "x^3-x^2-x+1=0") (3 1) nil 0.8)
activity: focus (affect: confusion - no prepared space for equation)
activity: implementation (affect: ok)
activity: length (affect: ok)
activity: complexity (affect: ok)
activity: motivation (affect: ok)
activity: temporary-focus (affect: ok)
((equation | x^3 - x^2 - x + 1 = 0|) (open) nil nil nil nil nil nil nil nil)
((alternative nil) 1 (6 4))
activity: length (affect: ok)
activity: focus (affect: ok)
((rule factor_out ((object (nil (x^2)))))) 1 2 nil nil t)
activity: focus (affect: ok)
activity: implementation (affect: ok)
activity: length (affect: ok)
activity: complexity (affect: ok)
activity: temporary-focus (affect: ok)
activity: motivation (affect: ok)
((rule factor_out ((object (nil (x^2)))))) nil nil nil nil nil nil nil nil (1) nil)
((pattern ((actor object) (object (((1 2) 4 term nil) nil))))))
activity: focus (affect: ok)
activity: temporary-focus (affect: ok)
activity: implementation (affect: ok)
((pattern ((actor object) (object (((1 2) 4 term nil) nil)))))) nil nil nil nil nil nil nil
  nil nil nil)
((expectation common_factor ((object ((2 2 term nil rest) nil))))))
activity: focus (affect: ok)
activity: temporary-focus (affect: ok)
activity: implementation (affect: disrespect - you put me in addition mode)
((expectation common_factor ((object ((2 2 term nil rest) nil)))))) nil ((like_before (6)))
  nil nil nil nil nil nil nil nil)
((result "x^2(x-1) - x+1=0") nil nil)
activity: focus (affect: ok)

```

activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result |  $x^2(x-1) - x + 1 = 0$ |) nil nil nil nil nil nil nil nil nil nil)  
 ((rule rewrite\_expression ((object (nil (-x+1))) (target (nil (|-(x-1)|)))))) 2 1 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((rule rewrite\_expression ((object (nil (-x+1))) (target (nil (|-(x-1)|)))))) nil nil nil nil nil  
 nil nil nil (2) nil)  
 ((result " $x^2(x-1) - (x-1) = 0$ ") nil nil t)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result |  $x^2(x-1) - (x-1) = 0$ |) nil nil nil nil nil nil nil nil ((realization expectation))  
 nil nil)  
 ((rule factor\_out ((object (nil (x-1)))))) 3 2 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((rule factor\_out ((object (nil (x-1)))))) nil nil nil nil nil nil nil nil (3) nil)  
 ((result " $(x-1)(x^2-1) = 0$ ") nil nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result |  $(x-1)(x^2-1) = 0$ |) nil nil nil nil nil nil nil nil nil nil)  
 ((rule factorization\_formula1 ((to (nil (x^2-1)))))) 4 1 nil nil nil)  
 activity: focus (affect: ok)  
 activity: implementation (affect: ok)  
 activity: length (affect: ok)  
 activity: complexity (affect: boredom - too easy)  
 activity: temporary-focus (affect: ok)  
 ((rule factorization\_formula1 ((to (nil (x^2-1)))))) nil nil nil nil ((easy -0.6)) nil nil nil  
 (4) nil)  
 ((result " $(x-1)^2(x+1) = 0$ ") nil nil)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 activity: implementation (affect: ok)  
 ((result |  $(x-1)^2(x+1) = 0$ |) nil nil nil nil nil nil nil nil nil nil)  
 ((continue) (product\_of\_factors) 0.7 t)  
 activity: focus (affect: ok)  
 activity: temporary-focus (affect: ok)  
 ((continue (product\_of\_factors)) nil nil nil nil nil nil nil nil nil nil)

## Output File of the Comprehension-Processes Module

- 1 - ((topic third\_degree) (open) nil nil  
((highlight\_attributes third\_degree ((interesting 0.8) (challenging 0.7)))) nil nil  
nil nil nil nil)
- 2 - ((method topic third\_degree t not teach) nil nil general nil nil nil nil nil negative)
- 3 - ((method topic third\_degree nil ((factor\_out) (factorization\_formula))) nil nil specific  
nil nil nil nil ((violation expectation)) nil nil)
- 4 - ((equation | x^3 - x^2 - x + 1 = 0) (open) nil nil nil nil nil nil nil nil)
- 5 - ((rule factor\_out ((object (nil (x^2)))))) nil nil nil nil nil nil nil (1) nil)
- 6 - ((pattern ((actor object) (object (((1 2) 4 term nil) nil)))) nil nil nil nil nil nil nil  
nil nil)
- 7 - ((expectation common\_factor ((object ((2 2 term nil rest) nil)))) nil  
((like before (6))) nil nil nil nil nil nil nil)
- 8 - ((result | x^2(x - 1) - x + 1 = 0) nil nil nil nil nil nil nil nil)
- 9 - ((rule rewrite\_expression ((object (nil (-x+1))) (target (nil (|-(x-1)|)))) nil nil nil  
nil nil nil nil (2) nil)
- 10 - ((result | x^2(x - 1) - (x - 1) = 0) nil nil nil nil nil nil nil  
((realization expectation)) nil nil)
- 11 - ((rule factor\_out ((object (nil (x-1)))))) nil nil nil nil nil nil nil (3) nil)
- 12 - ((result | (x - 1)(x^2 - 1) = 0) nil nil nil nil nil nil nil nil)
- 13 - ((rule factorization\_formula1 ((to (nil (x^2-1)))))) nil nil nil nil ((easy -0.6)) nil  
nil nil (4) nil)
- 14 - ((result | (x - 1)^2(x + 1) = 0) nil nil nil nil nil nil nil nil)
- 15 - ((continue (product\_of\_factors)) nil nil nil nil nil nil nil nil)

## Output of the Sentence Composer

Let us look at a challenging subject, namely third degree equations, which is also quite interesting. Unfortunately, we shall not go over a general technique for solving equations in this topic. Nevertheless, we can solve certain kinds of third degree equations by factoring out common factors, or, alternatively, applying the appropriate factorization formula. Here is an equation:

$$x^3 - x^2 - x + 1 = 0$$

First, since  $x^2$  is a factor common to the first and second terms, we factor it out. As you know, we perform this operation hoping to get a factor common to the rest of the terms. Through it we get the following result:

$$x^2(x - 1) - x + 1 = 0$$

Next, we rewrite  $-x+1$  as  $-(x-1)$ , arriving at the result we were hoping for:

$$x^2(x - 1) - (x - 1) = 0$$

Afterwards, we factor out  $x-1$ , yielding:

$$(x - 1)(x^2 - 1) = 0$$

Thereafter, we apply the factorization formula  $a^2 - b^2 = (a+b)(a-b)$  to  $x^2-1$ . This method is very simple. By means of it we arrive at the following result:

$$(x - 1)^2(x + 1) = 0$$

We obtain the solution by solving separately for each factor.

## REFERENCES

- Anderson J.R. (1980), *Cognitive Psychology and its Implications*, W.H. Freeman and Company, San Francisco.
- Anderson J.R. (1983), *The Architecture of Cognition*, Harvard University Press, Cambridge, Massachusetts.
- Appelt D. E. (1982), *Planning Natural Language Utterances to Satisfy Multiple Goals*. Technical Note 259, SRI International, March 1982.
- Bork, A. (1981), *Learning with Computers*, Digital Press.
- Brown, J.S., Burton, R.R. and DeKleer, J. (1981), Pedagogical, Natural Language and Knowledge Engineering Techniques in SOPHIE I,II and III. In D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, London: Academic Press, 1982.
- Brown, J.S. and VanLehn K. (1980), Repair Theory: A Generative Theory of Bugs in Procedural Skills. In *Cognitive Science*, 4, 379-426.
- Bundy, A. (1983), *The Computer Modelling of Mathematical Reasoning*. Academic Press.
- Burton, R.R. and Brown, J.S. (1981), An Investigation of Computer Coaching for Informal Learning Activities. In D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, London: Academic Press, 1982.
- Carbonell, J.G. (1982), Meta-Language Utterances in Purposive Discourse. Report No. CMU-CS-82-125, Carnegie-Mellon University, June 1982.
- Clancey, W.J. (1979), *Transfer of Rule-Based Expertise through a Tutorial Dialogue*. Doctoral Dissertation, Computer Science Department, Stanford University, California.
- Clancey, W.J. (1981a), Methodology for Building An Intelligent Tutoring System. Report No. STAN-CS-81-894, Stanford University, Stanford, October 1981.

- Clancey, W.J. (1981b), The Epistemology of A Rule Based Expert System: A Framework for Explanation. Report No. STAN-CS-81-896, Stanford University, Stanford, November 1981.
- Clancey, W.J. (1981c), Tutoring Rules for Guiding a Case Method Dialogue. In D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, London: Academic Press, 1982.
- Collins, A. (1976), Processes in Acquiring Knowledge. In Anderson R.C., Spiro R.J. and Montague W.E. (Eds), *Schooling in the Acquisition of Knowledge*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1976.
- Davey, A. (1979), *Discourse Production*. Edinburgh University Press, Edinburgh.
- Davis, R.B. (1974), What classroom role should the PLATO computer system play?, *AFIPS - Conference Proceedings*, Volume 43, pp. 169-173.
- Davis, R.B. et al. (1982), The Roles of "Understanding" in the Learning of Mathematics, Part II of the Final Report of National Science Foundation Grant NSF SED 79-12740, Curriculum Laboratory, University of Illinois, Urbana/Champaign, April 1982.
- Dunkin M.J. and Biddle B.J. (1974), *The Study of Teaching*, Holt, Reinhart and Winston.
- Dyer, M.G. (1982), *In-Depth Understanding, A Computer Model of Integrated Processing for Narrative Comprehension*. Report No. 219, Doctoral Dissertation, Department of Computer Science, Yale University, New Haven, Connecticut.
- Farnes N.C. (1973 revd. 1975), Comprehension and the Use of Context, Unit 4, Reading Development. Educational Studies: a Post-Experience Course and 2nd Level Course P.E. 261.
- Fikes, R. and Kehler T. (1985), The Role of Frame-Based Representation in Reasoning. In *Communications of the ACM*, pp. 904-920, Vol. 28 (9), September 1985.
- Forbus K. and Stevens A. (1981), Using Qualitative Simulation to Generate Explanations. Report No. 4490, Bolt Beranek and Newman, Cambridge, Massachusetts, March 1981.
- Genesereth, M.R. (1978), *Automated Consultation for Complex Computer Systems*. Doctoral Dissertation, Division of Applied Mathematics, Harvard University, Cambridge, Massachusetts.
- Genesereth, M.R. (1981), The Role of Plans in Intelligent Tutoring Systems. In D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, London: Academic Press, 1982.

- Goldstein, I.P. (1981), *The Genetic Graph: A Representation for the Evolution of Procedural Knowledge*. In D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, London: Academic Press, 1982.
- Grimes J. E. (1975), *The Thread of Discourse*. Mouton, The Hague.
- Grosz B.J. (1977), *The Representation and Use of Focus in Dialogue Understanding*. Technical Note 151, SRI International, Menlo Park, California, July 1977.
- Grosz B.J. (1979), *Focusing and Description in Natural Language Dialogues*. In A. Joshi, B.L. Webber and I. Sag (Eds.), *Elements of Discourse Understanding*, Cambridge University Press, 1981.
- Grosz B.J. and Sidner C.L. (1985), *Discourse Structure and the Proper Treatment of Interruptions*. In *IJCAI-85 Proceedings*, pp. 832-839.
- Hallyday M.A.K. and Hassan R. (1976), *Cohesion in English*. Layman Press, London.
- Hayes-Roth F., Waterman D.A., Lenat D.B. (eds) (1983), *Building Expert Systems*, Addison-Wesley.
- Hobbs J.R. and Evans D.A. (1980), *Conversation as Planned Behaviour*. In *Cognitive Science* 4, pp. 249-377.
- Hoey M. (1979), *Signalling in Discourse*. English Language Research, University of Birmingham, Birmingham Instant Print Limited.
- Knuth D.E. (1975), *The Art of Computer Programming, Fundamental Algorithms*, Vol(1), Addison-Wesley, publishers.
- Kukich K. (1983), *Knowledge-Based Report Generation: A Knowledge-Engineering Approach to Natural Language Report Generation*. Doctoral Dissertation, The Interdisciplinary Department of Information Science, University of Pittsburgh, Pennsylvania.
- Linde C. Goguen J.A. (1978), *Structure of Planning Discourse*. In *Journal of Social and Biological Structures*. Vol 1, pp. 219-251.
- Longacre, R. E. (1976), *An Anatomy of Speech Notions*. Peter de Ridder Press Publications in Tagmemics No. 3.
- Mann, W.C. and Moore, J.A. (1980), *Computer as Author - Results and Prospects*. Report No. ISI/RR-79-82, Information Sciences Institute, Los Angeles, January 1980.
- Mann, W.C. et al. (1981), *Text Generation: The State of the Art and the Literature*. Report No. ISI/RR-81-101, Information Sciences Institute, Los Angeles, and University of Pennsylvania MS-CIS-81-9, Philadelphia, December 1981.

- Mann, W.C. and Thompson S.A. (1983), Relational Propositions in Discourse. Report No. ISI/RR-83-115, Information Sciences Institute, Los Angeles, November 1983.
- Mann W.C. (1984), Discourse Structures for Text Generation. Report No. ISI/RR-84-127, Information Sciences Institute, Los Angeles, February 1984.
- Mann, W.C. and Thompson S.A. (1985), Assertions from Discourse Structure, Report No. ISI/RR-85-155, Information Sciences Institute, Los Angeles, April 1985.
- Matz, M. (1980), Towards a Computational Theory of Algebraic Competence. In *Journal of Mathematical Behaviour*, Vol. 3, No. 1.
- Matz, M. (1981), Towards a Process Model for High School Algebra Errors. In D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, London: Academic Press, 1982.
- McArthur D. (1984), A Graphical Interactive Tutorial Environment for Basic Algebra. Rand Corporation, Santa Monica, California, July 1984.
- McDonald, D.D. (1980), *Natural Language Production as a Process of Decision Making Under Constraint*. Doctoral Dissertation, draft version, MIT, Cambridge, Mass.
- McKeown, K.R. (1982), *Generating Natural Language Text in Response to Questions About Database Structure*. Doctoral Dissertation, The Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia.
- McKeown, K.R. (1985), Discourse Strategies for Generating Natural Language Text. In *Artificial Intelligence* 27, pp. 1-41.
- Miller P.L. (1983), ATTENDING: Critiquing a Physician's Management Plan. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol (5), No. 5, September 1983.
- Minsky, M. (1975), A Framework for Representing Knowledge. In P. Winston (Ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975, pp. 211-277.
- Quirk R., Greenbaum S., Leech G. and Starvik J. (1972), *A Grammar of Contemporary English*. Longman Group Limited, London.
- Reichman R. (1978), Conversational Coherency. In *Cognitive Science* 2, pp. 283-327.
- Reichman-Adar R. (1984), Extended person-machine interface. In *Artificial Intelligence* 22, pp. 157-218.



- Schank, R.C. and Abelson R.P. (1977), *Scripts Plans Goals and Understanding*, Lawrence Erlbaum Associates, publishers.
- Schank, R.C. and Riesbeck, C.K. (1981), *Inside Computer Understanding*, Lawrence Erlbaum Associates, publishers.
- Sleeman, D.H. (1981a), Assessing Aspects of Competence in Basic Algebra. In D. Sleeman and J.S. Brown (Eds.), *Intelligent Tutoring Systems*, London: Academic Press, 1982.
- Sleeman, D.H. (1981b), A Rule-based Task Generation System. In *IJCAI7-81 Proceedings*, pp.882-887.
- Sleeman, D.H. (1982), A Rule Based Modeling System. In *ECAI8 Proceedings, European Conference on AI*, pp. 160-164.
- Suppes, P. (1967), Some Theoretical Models for Mathematics Learning. In *Journal of Research and Development in Education*, 1, 5-22.
- Swartout W.R. (1982), XPLAIN: A System for Creating and Explaining Expert Consulting Programs, USC/Information Sciences Institute, Los Angeles, March, 1982.
- Sweet H. (1891), *A New English Grammar Logical and Historical*. Oxford, at the Clarendon Press, 1892.
- Waterman D.A. (1986), *A Guide to Expert Systems*, Addison-Wesley.
- Watkins A.E. (1979), *The Effect of the Symbols and Structures of Mathematical English on the Reading Comprehension of College Students*. Doctoral Dissertation, School of Education, University of California, Los Angeles.
- Wilkins J. (1668), *An Essay Towards a Real Character and a Philosophical Language*. London, 1668.
- Winter E. O. (1968), Some Aspects of Cohesion. In *Sentence and Clause in Scientific English*, by R. D. Huddleston et al., Communication Research Centre, Department of General Linguistics, University College, London, May 1968, pp. 560-604.
- Winter E. O. (1977), A Clause-relational Approach to English Texts: A Study of Some Predictive Lexical Items in Written Discourse. In *Instructional Science*, Vol. 6, No. 1, January 1977, pp. 1-92.
- Woods, P. and Hartley, J.R. (1971), Some Learning Models for Arithmetic Tasks and their Use in Computer-based Learning. In *British Journal of Educational Psychology*, 41 (1), 35-48.

- Woods, W. A. (1970), Transition Network Grammars for Natural Language Analysis. In *Communications of ACM*, pp. 591-606, Vol. 13 (10), October 1970.
- Zukerman, I. and Pearl, J. (1986), Comprehension-Driven Generation of Meta-technical Utterances in Math Tutoring. To appear in *AAAI Conference Proceedings*, August 1986.