Building Databases for the Computer-Based Memorization System

Richard C. Brandt
UUCS–88–002
Department of Computer Science
University of Utah
Salt Lake City, Utah 84112
January 18, 1988

Abstract

The Computer-Based Memorization System (CBMS) consists of eight games used by students to memorize groups of related facts that are stored in an associative network. The instructional design is built into the games. Designing and implementing associative networks for the CBMS is very different from the creation of traditional script-based Computer-Aided Instruction (CAI). The majority of the authoring effort is concentrated on the specification and representation of the factual knowledge to be mastered by students. This report describes the cognitive science basis for CBMS and aspects of authoring CBMS databases.

The associative networks used with CBMS have been modified for educational applications. First, numbers are associated with nodes and facts (two nodes connected by a link). These numbers permit the development of a model of the students' knowledge which permit the games to adapt to student performance. The numbers may be set by authors to emphasize important facts. Second, pictures may be associated with nodes. Third, groups of anticipated answers may be associated with nodes to allow ranges of acceptable student answers.

Building Databases for the Computer-Based* Memorization System

Richard C. Brandt
Computer Science Department
University of Utah
Salt Lake City, Utah, 84112

January 18, 1988

1 Introduction

Building databases for the Computer-Based Memorization System (CBMS) is different from creating ordinary script-based Computer-Aided Instruction (CAI). In script-based CAI, authors specify precisely the materials to be presented and the type of presentation. In CBMS, authors specify the facts that students are to know and how well these facts are to be known. In the domain of fact acquisition the creation of instructional materials using CBMS knowledge databases appears to be significantly less time-consuming than the creation of script-based CAI for the same purpose. In addition, the use of CBMS for memorization appears to be on a sounder theoretical basis than using ordinary script-based CAI.

CBMS uses an associative network for its knowledge database. An associative network is a knowledge representation that uses associations for its basic representation of knowledge. Objects in this representation are described by associating attributes with them. This paper describes the issues involved in building the files that are used to create an associative network. The associative network is used by the CBMS games.

1.1 CBMS Games

Students use eight CBMS games[1] to learn facts. The eight games use randomness and scoring to make the memorization more interesting. In addition, the games use and

*The development of the CBMS programs was supported in part by the Navy Personnel Research and Development Center Contract N00244-83-C-1759.

[1]The games are described in detail in the Computer-Based Memorization System Student Manual[4].

modify a file containing a model of the student's current knowledge of the material in the database. When the data in this file shows that a student knows a particular fact, the games stop asking questions about that fact.

The games distinguish between recall and recognition answers. Students are usually first given the opportunity to enter (recall) the answer to a question. If students are unable to recall the answer, they can try to select (recognize) the correct answer when it is displayed in a list of possible answers.

Although the main purpose of the CBMS games is the acquisition of factual knowledge, different games have different purposes that help students develop a variety of procedural skills. The particular purposes of the eight CBMS games are:

Concentration. Concentration's purpose is to reinforce the *perception of relationships* among items. The game uses a matrix board on which each row and column corresponds to items in a selected category.

Constraint. Constraint's purpose is to help students learn to *discriminate among members* of a selected category. Students decide whether a given statement is true or false for each item in a list of category members.

Flashcard. Flashcard's purpose is to help students *recall all facts* about members of a selected category. Flashcard is an electronic version of conventional study aids based on flash cards.

Identify. Identify's purposes are to help students (1) *identify members* of a selected category from short descriptions and (2) *review facts* about the identified members.

Jeopardy. Jeopardy uses a matrix game board to teach students *to identify members* of selected categories from their complete descriptions.

Matrix. Matrix's purpose is to help students learn those items in the database *have similar properties.*

Picture Quiz. Picture Quiz's purpose is to provide students with practice in *recognizing objects from pictures.*

Twenty Questions. Twenty Questions' purpose is to give students practice in *selecting and creating discriminating questions.* Students ask questions with yes/no answers to eliminate all but a single item from a list of database items.

1.2 Other CBMS Programs

In addition to the games, there are three other CBMS programs.

Browser. Browser is used by authors and students to examine CBMS databases. Browser shows the structure of a database and lists attributes of items in the database.

Fromtext. Fromtext is used by authors to convert text representations of associative networks into forms that can be used by the CBMS games and Browser.

Totext. Totext is used by authors to convert CBMS databases from the forms used by the

CBMS games to a text representation that can be examined and modified.

1.3 Cognitive Science Basis

The design of the CBMS games was influenced by recent work in Cognitive Science. The CBMS games are based on work on the Tactical Gaming System by Drs. Hollan, Crawford, and McCandless of the Navy Personnel and Research and Development Center[10,6]. The designs of both the Tactical Gaming System and CBMS were influenced by the associative net model of memory. The design of CBMS was also affected by experiments on recall and recognition. The associative net models and recall experiments are described in this section.

1.3.1 Associative Net Memory Models

The associative net models of memory provide the theoretical basis for CBMS.[2] Variants of this memory model have proved useful in describing memory functions such as retention and retrieval.

The associative net model of memory[3] asserts that memory is constructed from elements (words, strings, pictures) interconnected to form cognitive units called traces. For example, assume the following fact is in long-term memory.

```
"Ronald Reagan" "was elected" president;
```

The three phrases

```
"Ronald Reagan"
"was elected"
president
```

are called *elements*, while the interconnection of the three phrases that forms a fact is called a *trace*. Each element can be involved in many traces. For example, most people can recall many traces (facts) involving each of three elements above.

In the associative net model, if a trace or element in long-term memory is to be recalled, there must be some connection path to it through traces from elements in working memory. For example, if a student is asked to name the last five presidents, and the trace "Ronald Reagan was elected president" is in the student's long term memory, there is a certain probability that the student will recall the element "Ronald Reagan" because there is a connection from the element "president" to the element "Ronald Reagan." In other words, since the trace "Ronald Reagan was elected president" is in long term memory of the student, the student may *associate* the element "Ronald Reagan" with the element "president."

[2] Associative nets have been used for describing memory since the initial work of Collins and Quillian.[5] They have also been used for substantially more complex systems. For example, Rumelhart and Norman[12] treated networks including nodes for events, concepts, and episodes. *Associative Networks* edited by Findler[8] contains a collection of essays on associative networks.

[3] Much of the information discussed in this section comes from John R. Anderson's book, *Architecture of Cognition*.[1]. The model discussed here closely follows the model used in Anderson's ACT* theory.

In the associative net model, if an element or trace is in working memory and not yet in long-term memory, there is a certain probability (this probability depends on the individual) that a copy of the element or trace will be created in long-term memory. Furthermore, if an element or trace is in both working memory and long-term memory, the *strength* of the element or trace in long-term memory is increased. The greater the strength of an element or trace, the higher the probability the element or trace will be recalled. Additional separate occurrences in working memory of the element or trace increase further both the strength of the element or trace and the recall probability. If a element or trace is not used, its strength slowly decreases with time at a rate that is dependent on the individual.

The probability that a trace or element will be recalled is related to its *activation*. Activation is a measure of the "activeness" of a trace or element. The greater the activation, the higher the probability that an item will be recalled. The initial sources of activation are items and images in working memory that have been received from the external environment. The activation spreads from an active mode to its nearest neighbors (closest associates). For example, if the element "president" is activated, activation will spread to the elements "Reagan," "Carter," "Ford," "Nixon," "Johnson," and "Kennedy." Each of these associates will be activated; the relative activation of each element is proportional to its strength. For example, if each of the elements has equal strength, each will be activated equally; or if an element has much greater strength than the other elements its activation will be much greater than that of the other elements. The amount of activation that an element can provide is limited and independent of the number of its closest associates. For example, call the amount of activation "A." If an element has just one associate, it will receive the entire activation "A." If an element has five associates each of equal strength, each will receive a fifth of the activation, "A/5." This is called the *fan* effect.

The activation will spread through the associative network from neighbor to neighbor. An element or trace may receive activation from several neighbors. For example, assume a student is asked to name the man "who was elected president in 1800 and wrote the Declaration of Independence." Then the elements in working memory that serve as sources of activation are:

```
president
1800
"Declaration of Independence"
```

These all are associates (nearest neighbors) of the element "Jefferson" and each will contribute to the activation of the element "Jefferson."

1.3.2 Recall and Recognition

Recall and recognition experiments have demonstrated that the length of time that a trace is continuously in working memory has little effect on its later recognition or recall.[11] Consequently, there is no point in having

students repetitively answer the same question in the CBMS games. A two-in-a-row game rule reflects this. If a student answers the same question correctly twice[4] consecutively,[5] the game will not ask the same question again.

Recall answers are generally more difficult for students for two reasons. First, in the associative net model, there is less activation for a recall answer than a recognition answer since one less component of a fact is shown to the student. Second, recall answers in games require that students type the answers. Because of these added difficulties, the CBMS games always give more points for recall answers when both types of answers are used.[6]

Past experiments have shown that practice leads to greater accuracy and shorter recall times. This is particularly true if the practice sessions are separated by some other activity or by time. For example, Anderson[1] (p. 183) found that the recognition time for sentences consisted of two components: an intercept time independent of the number of practice sessions and a component inversely proportional to the number of practice sessions to the 0.36 power; that is

$$T = Intercept + Constant \star P^{-0.36}$$

where P is the number of practice sessions. In CBMS use of the two-in-a-row rule and an "Importance" file permits authors to control the minimum number of separate practice sessions in which students will have to answer questions involving a database element or trace correctly. This is achieved in the following manner. Associated with each element and each trace in a CBMS database is a number called "Importance." Each time a student answers a question involving a particular element or trace correctly, one is subtracted from the Importance of the element or trace; each time a student answers a question incorrectly, one is added to the Importance. The probability that a particular element or trace will appear in a question is proportional to its Importance. When the Importance of an item or trace reaches zero, it is not used in questions. The two-in-a-row rule keeps the Importance of an item or trace from going down more than two during any game session. This allows an author to specify in advance the minimum number of practice sessions that will include a particular trace or element. For example, if an author initializes the Importance of an element or trace at 10, the students will see questions involving the element or trace in at least five ($5 = 10/2$) separate sessions.

1.4 CBMS Databases

CBMS databases are associative networks that are used to help students memorize facts. As such, they contain statements and items. Statements correspond to traces in memory and items correspond to elements in memory.

[4] Nelson[11] demonstrated a small but not insignificant positive effect of repetition within a test session, particularly if there were intervening questions.

[5] There may be, and almost always are, intervening questions in a game.

[6] During a game students may obtain more points by selecting recognition answers simply because it takes less time to move the cursor to a correct answer in a menu than to type a recall answer.

1.4.1 Statements

CBMS databases allow two types of statements:

- Categorization statements. A categorization statement specifies the category to which an element belongs. An element may belong to only one category.
- Attribute statements. An attribute statement specifies a property of an element in the databases.

Categorization probably plays a more important role in CBMS than it does in memory. The category membership statement in CBMS is "isa." Categories are important in CBMS games because to play students select one or more categories for study. Categorization in CBMS is stricter than it is in memory. Each item in CBMS must belong to one and only one category. For example, CBMS prohibits having the following two statements in the same database.

```
"Ronald Reagan" isa Republican;
"Ronald Reagan" isa president;
```

The category structure in CBMS is similar to biological classification including inheritance of attributes. For example, assume the following three classification statements are in the database.

```
hound isa dog;
beagle isa hound;
Snoopy isa beagle;
```

In CBMS "Snoopy" will inherit both the attributes of dogs, hounds, and beagles. For example, "Snoopy"will inherit the attributes given in the following statements:

```
dog has "4 legs";
dog eats meat;
dog "related to" wolf;
hound ears drooping;
hound voice deep;
hound "follows prey by" scent;
beagle coat smooth;
beagle legs short;
beagle size small;
```

Because of inheritance[7], categories may only have attributes that characterize all or a least most of their members. Because of inheritance, authors need to associate directly with a given item only those attributes that distinguish it from other members of the same category.

1.4.2 Items

Four types of items are permitted in CBMS:

- Words. A word is a single word.

[7] In CBMS, authors can prevent the inheritance of particular attributes. For example, it is possible to state that "birds can fly" and then prohibit penguins from inheriting that attribute.

- Strings. A string consists of an *ordered* list of words. For example, the phrase "Ronald Reagan" is a string.

- Pictures. A picture may contain graphics, text, and video.

- Alternative Answers. An alternative answer is a list of words and formatting statements that define acceptable student answers associated with an item. Alternative answers are used only when students' recall answers are being analyzed.

1.4.3 Constraints

CBMS databases are not as expressive as many other associative networks. Specifically, it is limited to three-part statements. For example, it is not possible to express in CBMS the following:

```
Snoopy followed Molly "to the park";
```

In a large part, this lack of expressiveness is because CBMS was designed to help students memorize factual knowledge. To do so, the CBMS games generate questions. Question generation is simple using three-part statements,[8] but it is difficult with four-part statements.

[8]Authors obtain questions with a particular form by associating question templates with the "relations" in a CBMS database.

Another constraint on CBMS databases is that in the games, students are often asked to recall an answer that is one part of a three-part statements. If the part is too complicated, it will be difficult for students to correctly recall and enter it. CBMS works better with short answers than with long ones.

2 Specifying the Knowledge

Specification of the facts to be mastered using a CBMS database is similar to ordinary instructional design.

2.1 Objectives

The first step in creating a CBMS database is to define the objectives and the intended audiences. What are students expected to know? This definition process is far more specific and restrictive for CBMS than for ordinary instruction. For example, the objective of a beginning physics course might be mastery of elementary mechanics. In the instructor's mind this includes both procedural and factual material. This type of specification is too general for CBMS. A more acceptable specification would be "the names and attributes of around 40 basic formulas in elementary mechanics and the names and attributes of around 10 problem-solving procedures.[9]" The expected audience might be

[9]CBMS requires that both everything be named and that everything be explicit. For example, this will require for a problem-solving procedure that it be named, that the criteria for using it be specified, and that its at-

specified as university freshmen and sophomores who have already taken one course in calculus. Another example of a specification acceptable for CBMS would be "200 facts on cranial nerves for first-year medical students in a neuroanatomy course."

Initial objectives may not be too specific because instructors often don't know how many facts they are currently presenting to students. However, setting an initial limit on the number of facts is needed to avoid the problem of too large databases.

2.2 Data Collection

Facts for use in a CBMS database may be obtained from various sources. Example sources are:

- textbooks
- reference books and user manuals
- teachers and instructors
- experts

In the process of collecting facts, the objectives will become more specific. Usually, the items that will be subjects of questions will become well defined. For each item, there will be a list of attributes that characterize it.[10] The initial lists of attributes probably will include facts that may not seem absolutely necessary for the student to know. These facts should be included initially but marked for possible later exclusion. The initial lists should not include facts that students clearly do not need to know.

A "naming" problem may occur at this stage if there are several names for an item or there is no name for an item. If there are several names for an item that students are expected to know, the names may be combined. For example, the first cranial nerve is called both "Olfactory Nerve" and "Cranial Nerve I." The two names may be combined to form "Olfactory N./CN I." The more difficult problem is the lack of any particular name for an item. For example, if the item is a particular problem-solving procedure in physics, the procedure may have no name. Two solutions are possible; either a place-holding name such as "ProcedureC" may be used (this defers the problem) or a name may be made up. Since CBMS permits phrases as names, the made-up name may be descriptive.

2.3 Common Features

Usually, after the initial set of facts is collected features common to many items will appear. For example, most cranial nerves:

- have at least one function
- enter the cranium through a foramen
- have at least one functional component
- have cell bodies in a particular location

tributes be listed. This is unlike the standard "example-driven" instruction in physics where an instructor will show how a problem is solved, but will not name the problem-solving procedure or explicitly identify the criteria for its use. Often students are expected to "discover" these criteria by doing homework problems.

[10] In CBMS, items cannot be distinguished by name alone.

Common features of database items are easier to spot if the attributes of each item are listed immediately after it in alphabetical order. Errors of omission can be detected by reviewing the common features. For example, a review of a database may show that many but not all items have certain features. Should all items, or at least all similar items, have these features? If they should, then more data must be collected.

2.4 Categorization

The categorization of items when the subject matter has an already defined structure is easy; the categorization can follow the existing structure. For example, most biological systems are already classified. When there is no usable predefined structure, the items may be categorized using the following rules.

1. Items with common features belong in the same category.
2. A category usually should have only features common to its members.
3. The categorization should be shallow; that is, a student selecting a category to study should not have to make more than one or two choices.

2.5 Expert Review

After the subject matters has been organized into a database, it should be reviewed by other subject matter experts. Generally, they will address the following questions:

- Are there items omitted that should be included and vice versa?
- Are there facts (statements) omitted that should be included and vice versa?
- Is the categorization reasonable or are there better categories?

Some disagreement is to be expected. In particular, there is often disagreement between individuals who use the information in the field and those who teach it.

2.6 Revision

After experts have reviewed the database, it must be revised. Before making the revision, an author should decide how many facts the students need to know. Authors sometimes make the mistake of including in the database all facts that the experts think are important. This ignores the fan effect, which implies that the more attributes an item has, the more difficult it will be to recall any particular one.[11]

It is important to identify and include those facts that experts use in item recognition and in decision-making. A recent study by Grant and Marsden on medical diagnosis[9] revealed that differences in diagnostic expertise between specialists and medical students could be explained by identifying facts that the specialists felt were most important ("forceful facts") in the diagnosis.

[11] The fan effect states that if all attributes of a particular item have equal strength, the activation is proportional to some value A divided by n, where n is the number of attributes.

3 Implementation

Implementing a CBMS database includes four steps:[12]

- creating and editing the defining text file with an ordinary text editor
- creating and editing pictures with the sequence editor
- compiling the files with Fromtext
- evaluating the databases with Browser and the game programs

3.1 CBMS Text Files

CBMS text files may be edited and created with any text editor that produces an ASCII text file.

3.1.1 Basic Forms

The basic categorization and attribute statements in a CBMS text file consist of three parts:

1. subject
2. relation
3. object

Each part may be a word or a string. A string is just an ordered collection of words or numbers; for example, the phrases "Ronald Reagan" and "4 wheels" are strings. Strings are marked at the beginning and end with double quotation marks. A statement is always ended with a semicolon. Example statements are:

```
"Ronald Reagan" "was elected" president;
"abducens nucleus" isa nucleus;
```

Occasionally the second part of a statement, the relation, corresponds to a verb.

Categorization All items in CBMS databases are categorized. An item belongs *directly* to a category if there is a categorization statement of the form:

```
item_name isa category_name;
```

An item is an *indirect* member of a particular category if it is a member of another category that is either a direct or indirect member of the particular category. There are four predefined categories; the most general category to which all items in the database indirectly belong, *db_item* and the three categories:

```
db_element
db_relation
db_object
```

All items that are to be used as subjects of questions must belong either directly or indirectly to the category *db_element*. A member of the category *db_element* may be used as either the subject or object in

[12] Implementation of CBMS databases is described in detail in the Computer-Based Memorization System Author and Instructor Manual[3].

statements. Items that may be used as either the subject or the object in statements but *not* as the subjects of questions are placed in the category *db_other*. Only members of the category *db_relation* are used for relations.

Attributes Attributes are statements that describe items in the database. Any statement that does not use the *isa* relation is an attribute statement. The complete description of an item consists of attributes specific to an item and attributes that characterize the categories to which the item belongs. For example, the attributes specific to a beagle are:

```
beagle coat smooth;
beagle legs short;
beagle size small;
```

Beagles also have the attributes of hounds and dogs since they are members of the categories hounds and dogs. Note that the preceding statements are terse and the relations in the statements restrict the posssible objects. In CBMS, the question formats are associated with the relations. Possible questions that might be associated with the preceding three statements are:

```
What type of coat does a beagle have?
What length (short, medium, long) legs
  does a beagle have?
How large a dog is a beagle?
```

If a more general relation such as "has" is used, the questions cannot be as specific. For example, assume the attributes specific to a beagle are:

```
beagle has "smooth coat";
beagle has "short legs";
beagle has "small size";
```

Then only one general question is possible.

```
What does a beagle have?
```

This question with its three answers is more difficult than the preceding three specific questions.

The games use relations in hint generation. The alternatives presented in the hint list are other objects of a relation. For example, other objects of the specific relation "coat" might be: "heavy," "rough," "short," "wire-haired," and "long." The use of closely related alternatives is often desirable. The generation of these alternatives depends on the use of the same relation. For example, if the relation "hair type" is used sometimes and the relation "coat" also, the objects of "hair type" will not be used as hints for an object of a relation using "coat."

A further advantage to the use of the same relation when possible (for example, rather than using "coat" and "hair type" using only "coat") is that its use helps create alternative paths to an object in the mind; these help students remember objects. For example, if students cannot remember the type of coat a beagle has they might think that a beagle is like a fox hound and has a similar coat.

If there is no problem with ambiguity, relations can be terse. For example, "coat" is a terse relation. Terse relations require less typing and

often are easier to locate in a database. However, terse relations may result in a database that is hard to read and to evaluate.

3.1.2 Organization

It is easier to review a database that has a consistent structure. The alphabetical outline method is one reasonable way to organize an database. In this method, all items belonging to *db_element* are placed first, all items belonging to *db_other* are placed second, and all items belonging to *db_relation* are placed last. For each major category, the items and categories that are its direct members are placed in alphabetical order. Finally, the attributes of each item are listed in alphabetical order. Indentation similar to that used in ordinary outlines may be used.

If there is some natural order, then this should be used when appropriate. For example, the twelve cranial nerves are listed in numerical order.

Categories in the database should not have too many direct members. There are two problems with having too many direct members; first, the fan effect will reduce the activation of each member and second, impossible and unfair questions may be generated. For example, one experimental database resulted in the instruction:

```
Name the Democrats in Congress.
```

3.1.3 Sentence Templates

Sentence templates are patterns used to convert attribute and classification statements into forms that can be used in the games. They are used because questions directly constructed from attribute or classification statements may sound awkward or be difficult to understand. Sentence templates are associated with the relations in a database. There are four types of sentence templates:

- Statement
- Subject
- Object
- Yes_No

Statement templates are used to convert database statements to acceptable English. For example, a Statement template may be used to convert the database statement, "beagle coat smooth," to "The coat of a beagle is smooth."

Subject templates are used to create questions when the answer will be the subject of a database statement. For example, a Subject template may be used to create the question "What type of hound has a smooth coat?" from the database statement, "beagle coat smooth."

Object templates are used to create questions when the answer will be the object of a database statement. For example, an Object template may be used to create the question "What type of coat does a beagle have?" from the database statement, "beagle coat smooth."

Yes_no templates are used to create questions

with yes or no answers from a database statement. For example, a Yes_no template may be used to create the question "Does a beagle have a smooth coat?" from the database statement, "beagle coat smooth."

Because sentence templates are associated with relations, they cannot explicitly mention either the subject or object of a particular database statement. Instead special symbols are used to represent the three general components of database statements.

"%s%" stands for the subject in a database statement, except when it is used in a Subject template. When %s% is used in a Subject template, it stands for the category to which the subject of a database statement belongs.

"%o%" stands for the object in a database statement, except when it is used in a Object template. When %o% is used in a Object template, it stands for the category to which the object of database statement belongs.

"%r%" stands for the relation in a database statement. Since sentence templates are associated with relations, %r% is used only when sentence templates are shared by several relations. Only sentence templates associated with a category of relations may be shared. Templates associated with a category of relations are "inherited" and hence shared by the category members. For example, if the category "dog feature" has the members "coat," "muzzle," and "tail," these members will share sentence templates associated with "dog feature."

Examples of sentence templates that may be associated with the relation "coat" are:

```
coat STATEMENT "The %r% of a %s% is
  %o%.";
coat SUBJECT "What has a %o% %r%?";
coat SUBJECT "What type of %s% has
  a %o% %r%?";
coat OBJECT "What type of %r% does
  a %s% have?";
coat YES_NO "Does a %s% have %o% %r%?";
```

These sentence templates will generate the example sentences and questions on beagles contained in this section (p. 12) when "beagle" is substituted for %s% (in the second Subject template, "hound" is substituted for %s%), "coat" is substituted for %r% and "smooth" is substituted for %o%.

If a database is to be used with all the games and the Browser, each of the four types of sentence templates must be associated directly or indirectly through inheritance with each relation in the database.

Creating Templates A sentence template for a question is formed by (1) creating a question from a database statement and (2) replacing the instances of the subject in the question with %s% and the instances of the object with %o%. Statement templates are created in a similar manner. Questions must be carefully written so that they are not ambiguous. Students generally regard ambiguous questions as unfair. Since the templates are associated with relations, the relations do not have to appear in templates. Often using a words somewhat different from the words in the relation will result in a clearer

question or statement. If questions or statements cannot be made clear for all potential objects or subjects, it may be necessary to introduce additional relations.

The articles used in the templates must be chosen with care. If none of the objects or subjects that appear after an article either start with a vowel or are plural, use of the article "a" is acceptable; otherwise, it is better to use the article "the."

3.1.4 Anticipated Answers

An **anticipated answer** may be specified by using the MATCHED_WITH relation in a statement. For example, the statement:

```
"Oculomoter N./CN III" MATCHED_WITH "CN
  III | oculomotor | cranial nerve III";
```

associates the anticipated answer "CN III | oculomotor | cranial nerve III" with the database item "Oculomotor N./CN III." Students' answers are first compared exactly against the item's name ("Oculomotor N./CN. III" in the preceding example). If a student answer matches exactly, it is marked correct. If a student answer does not match an item's name and an anticipated answer has been specified, the student answer is compared with the anticipated answer. If they match, the student answer is marked correct.

The use of anticipated answers makes the games much more acceptable to students and distinguishes CBMS databases from ordinary associative nets that do not use anticipated answers. Anticipated answers and the ways in which they are matched against student answers are described using the following terms.

Tokens Anticipated answers are constructed from special symbols and tokens. A token may be a number or an identifier. *There must be a space on both sides of a token unless it is the first or last token in the anticipated answer.*

A **number** is a sequence of digits that may include a few special characters. Examples are: "122," "1000, " and "1.34e-15." The default tolerance is 3%. For example, if an anticipated answer is 100, student answers between 97 and 103 will match. A different tolerance is specified by including after a number a percentage. For example, an anticipated answer of 100 %1 will match student answers between 99 and 101, inclusive. A range may be specified instead. A range is specified by a lower bound followed by two dots (periods) and then an upper bound. For example, if the anticipated answer is 101 .. 103, student answers between 101 and 103 inclusive will match.

An **identifier** is any sequence of letters and digits that is not a number. Examples of identifiers are: "ball," "operation," and "1ff10d." Identifiers in anticipated answers may include a spelling tolerance, specified by a tolerance percentage. The default tolerance is 20%. A different tolerance may be specified by placing a percent symbol after the answer followed by the tolerance percentage. A 0% tolerance requires exact matches. A spelling checker computes a score when it compares

two words; the score is then compared with the tolerance. Different errors are given different weights. For example, the initial letter of each word counts twice as much as the other letters and a transposition of characters counts as one third of a letter that disagrees. The spelling checker will remove the last "s" from a student answer if doing so improves the match. Examples of anticipated identifiers are: "trip," "helicopter %0," and "gadhafi %50."

Modifiers Modifiers are special symbols used as prefixes for tokens in anticipated answers so that student answers will be interpreted in special ways. There are three modifiers.

@c means that the cases of the letters (upper-case or lower-case) in a student's answer are important. Normally the cases are ignored. For example, the answers trip, Trip, and TRIP match the anticipated identifier trip. If the anticipated identifier is @cTrip %0, then only the answer Trip will match it. A letter in the wrong case counts as a letter that disagrees.

@l means that the number following @l is to be interpreted literally rather than as a number. For example, only the answer 5 will match @l5; the answer 5.00 will not. This format is useful in counting and with numbers used for identification rather than as numbers; for example, Boeing @l737.

@p means that the token is to be used as a pattern rather than as a separate token. A word will match the token if a portion of the word has the same pattern. For example, the pattern @pamp will match the student answers amps, amperes, amp, damp, kiloamps since in each case the pattern amp is present.

There is no space between a modifier and the token it modifies.

Words A **word** is a token or a token prefixed with a modifier. Examples of words are: "helicopter" and "@cWashington."

Wordlists and Match Answers A word list can be a single word or a list of words separated by spaces. Examples of word lists are:

```
five @l5
a the
helicopter
```

Word lists are used in Or anticipated answer parts and Optional match anticipated answer parts.

An Or answer part consists of the special symbol **@o** and a word list inside parentheses. The Or part of an anticipated answer is matched if the student's answer contains one item in the word list.

The Optional answer part consists of the special symbol **@?** and a word list inside parentheses. An example is "@?(a the)." The optional part matches if one or none of the words in the word list is in a student's answer.

An example of an anticipated answer containing an Or and an Optional part is: "I

like @?(a the) @o(red white) rose." It matches the following student answers:

```
I like red roses.
I like the red rose.
I like a white rose.
```

Since the spelling checker removes the last "s" from tokens if that will improve the match, the first student answer matches although it contains "roses" instead of "rose."

Range and Anything Match Answers

Anything match answers, **@a** answers, are used when the particular tokens in the student answer are not important; only the number of tokens is important. Three methods for specifying ranges are supported with "@a": an asterisk, *, specifies a range of zero or more tokens; a plus sign, +, specifies a range of one or more tokens; and two numbers separated by a comma, **m, n**, specify a range explicitly. For example, the anticipated answer "a big @a* ball" will match the following student answers:

```
a big ball
a big red ball
a big red and white ball
```

The tokens "red", "and", "white", match @a*. An example of the use of an explicit range is "a big @a1,3 ball" It matches the first two but not the second two of the following four answers:

```
a big red ball
a big red and white ball
a big ball
a big red white and blue ball
```

Part A part of an anticipated answer may be a word, an Anything match of a range of words, an Or match of one word in a word list, or an optional match to one word in a word list. For example, in the anticipated answer "I like @?(a the) @o(red white) rose" the parts are:

```
I
like
@?( a the )
@o( red white )
rose
```

Partlist A partlist is a list of one or more parts. An example of a partlist is:

```
I like @?( a the ) @o( red white ) rose
```

A partlist is matched if each of its parts match corresponding parts in the same order in the student answer.

Pattern A pattern is a partlist that may be marked at the beginning with **@b** or at the end with **@e**. If a partlist is marked at the beginning with @b the text after @b must be at the beginning of a student answer. If a partlist is marked at the end with @e, the preceding text must be at the end of a student answer.

For example, the anticipated answer

```
@b I like @?( a the ) red rose
```

will match the student answers:

```
I like red roses and purple violets.
I like the red rose that won first
  place in the contest.
```

but not the answers:

```
In the park, I like the red roses best.
I like violets and I like red roses.
```

If neither @e nor @b is present, the *partlist may appear anywhere in the student's answer.* If both @e and @b are present, a student answer containing the anticipated answer part alone is the only matching answer.

Excludes Occasionally, it is useful to look for the absence of some pattern in a student answer. To exclude a pattern, precede the pattern with the minus sign, -. For example, the following anticipated answer

```
@?( a the ) red rose -@o( not @pn't )
```

will match the following student answers:

```
I like red roses
he does like red roses
```

but not:

```
I do not like red roses
he doesn't like red roses
```

The excluded wordlist "not @pn't" provides a way of handling negatives in student answers. It only handles single negatives; it fails on double negatives.

Patternlist A patternlist is a list of patterns separated with the special symbol "|" and optionally followed by a group of patterns that are excluded. For example, the anticipated answer

```
@?( a the ) red rose -@o( not @pn't )
```

is a patternlist because it contains an excluded pattern. The first pattern is

```
@?( a the ) red rose
```

and the second pattern (which is excluded) is

```
@o( not @pn't )
```

Technically, all anticipated answers are patternlists; usually, they will consist of only one pattern. An example of a patternlist using the special symbol "|" is

```
He likes red roses | She likes violets
```

Note that there must be a space on both sides of the special symbol "|". "|" separates the patterns in a patternlist. Students' answers are matched if they match a pattern in the patternlist.

3.1.5 Changing Default Values

To simplify editing, Fromtext uses several defaults. The values of the defaults can be changed or overridden by including appropriate statements in a CBMS datafile.

Default Tolerances The default tolerances for numbers (3%) and strings (20%) may be changed. The change will affect following items in the file. For example, the statement

```
DEFAULT_NUMERICAL_TOLERANCE 10;
```

changes the default numerical tolerance from its current value to 10%. The statement

```
DEFAULT_STRING_TOLERANCE 15;
```

changes the default string tolerance from its current value to 15%. Default tolerances may be changed at several places in a CBMS datafile.

Specifying Importances The Importance of an item or a statement specified in the database text file determines the *net* number of times student must correctly answer questions involving a particular item or statement before that item or statement is considered known. When an item or statement is known, the games do not ask questions about it. The net number is the number of correct answers minus the number of incorrect answers. For example, if the statement:

```
beagle coat smooth IMPORTANCE 10;
```

is included in a CBMS database text file, the games will continue to ask questions involving the statement until the number of correct answers to questions involving the statement entered by a student exceeds the number of incorrect answers by 10.

The Default Importance used by Fromtext is 3. This value is used when the Importance of an item or fact is explicitly specified as above. The value of the Default Importance is changed by including a statement specifying a new value. For example, the following statement changes the Default Importance to 10:

```
DEFAULT_IMPORTANCE 10;
```

The Importance of the following items and facts in the CBMS databases will be 10 unless specified otherwise.

Inheritance The games use attribute inheritance, that is members of a category by default "inherit" the attributes of the category. For example, if beagles are specified with the "isa" statement to belong to the category of hound, beagles "inherit" the attributes of hounds (drooping ears, deep voice, etc.). Sometimes, it is desirable for a category to have attributes that are inherited by not all its members. This may be done by placing the word "NO_INHERITANCE" after a statement of the attribute that would normally be inherited by a category member. For example, the following statements

```
bird ability fly;
penguin isa bird;
penguin ability fly NO_INHERITANCE;
```

assert that all members of the category birds except penguins have the ability to fly.

Synonyms SYNONYM_FOR is a predefined relation that associates either a word with its synonym or a unit with a number and a related unit. Examples are:

```
in SYNONYM_FOR "1 inch";
in SYNONYM_FOR "2.54 cm";
feet SYNONYM_FOR "12 inch";
foot SYNONYM_FOR feet;
yard SYNONYM_FOR "3 feet";
meter SYNONYM_FOR "100 cm";
```

When the games are run, conversion tables are set up to use the values specified with the SYNONYM_FOR statements. These are used to convert units appearing in student answers. For example, if the anticipated answer is "1000 yards" and a student enters "3000 feet," the student's answer will match the anticipated answer if the preceding SYNONYM_FOR statements are in the database.

Hidden By default the games will display all possible questions. Occasionally it is undesirable to use particular items or relations in questions. This most often occurs with the "isa" relation because the categorization information can result in trivial questions. For example, in a database on cranial nerves, the question "What is the abducens nerve?" appears with the trivial answer of "cranial nerve" because the database contains the categorization statement:

```
"Abducens N./CN VI" isa "cranial nerve"
```

Questions of this sort can be eliminated by including the following statement in the database:

```
isa HIDDEN;
```

Placing the word "HIDDEN" after a relation or an item, "hides" it. The hidden relation or item will not be used in questions or in statements.

3.2 Pictures

Pictures including graphics, video, and text may be associated with items in the database. Pictures are created and edited as *sequences*. A sequence is a linear sequence of text, graphics, and video frames. In CBMS datafiles, a picture is identified by naming the file containing the picture and then associating the picture with an item. A file is identified by including a statement consisting of the word "USE_DISPLAYFILE" and the name of the file containing the pictures. Since pictures are always stored in sequence files, the file name extension, ".SEQ," is omitted. For example, the file "nerves.SEQ" is identified by including in the datafile the following statement:

```
USE_DISPLAYFILE nerves;
```

Pictures are associated with items by including a statement consisting of the word "USE_DISPLAY" and the name of the sequence that is the picture. For example, the sequence "abducens" is associated with the item "Abducens N./CN VI" by including in the datafile the following statement:

```
"Abducens N./CN VI" USE_DISPLAY
   abducens;
```

This statement causes Fromtext to copy the sequence "abducens" from the file that was specified with USE_DISPLAYFILE. Several files may be specified with USE_DISPLAYFILE; Fromtext copies sequences from only the most recently specified file.

The Sequence Editor,[13] **se**, is used to create and edit sequences. The Sequence Editor is started by entering in response to the system prompt "sc" and the name of the file to be edited or created. For example, to edit or create the file nerves,

```
se nerves
```

is entered in response to the system prompt. The screen will be cleared and the Identification Menu (see Fig. 1, p. 20) will be displayed.

The Identification menu lists the sequences in the file and the "parent" of the file that is represented by two dots. A sequence is created by choosing the command Create and then entering its name. A sequence is edited by moving the cursor to the sequence's name (use the up and down arrow keys or CTRL P and CTRL N) and then choosing the command Select. After a sequence is selected or created, the screen is cleared and the Sequence Instruction Menu (see Fig. 2) will be displayed.

The Sequence Instruction Menu lists the sequence instructions. A sequence instruction usually consists of a command that identifies

[13]The Sequence Editor is described in detail in the Sequence Editor Manual[2].

FILE: /cai/cbess/interaction/nerves
Targets: (frames) Sequence, Display

Identification

1.	..	<Parent>
2.	abducens	<Sequence>
3.	oculomotor	<Sequence>

Select from Above

Identify: Inspect Select Create Copy Link_to Quit

Figure 1: Identification Menu

the type of frame and the frame name. For example, the instruction

```
Show_graphics "abducens highlight"
```

specifies that the graphics frame named "adbucens highlight" is to be displayed. The commandline at the bottom of the screen lists the commands. Only the following three commands are described here: Edit, Graphics, and Video. An instruction is edited by moving the cursor next to the instruction and then choosing the command Edit. A graphics or video instruction is created by choosing the command Graphics or Video and then entering the name for the graphics or video frame.

Graphics When a graphics frame is edited or created, the screen is cleared and the main graphics commandline along with the graphics is displayed (see in Fig. 3).

Authors use the graphics command Edit to

Sequence: abducens

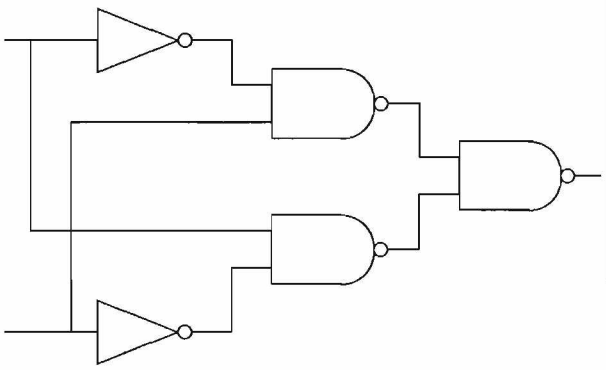
Sequence Instructions

1. Show_video "abducens"
2. Show_graphics "abducens highlight"
3. Wait System
4. End of List

Select from Above

Seq.: **Edit** Window Line Menu Graphics Video Note Clear Prompt Wait Backup Name Tryout >>

Figure 2: Sequence Instruction List



FRAME: **Edit** Undo Working View Name Clear Quit Status Modify Reorder Copy Examine >>

Figure 3: Graphics Editing

create and edit graphics. When the Edit command is selected, the Edit commandline shown below is displayed. This commandline lists the graphics objects that may be used:

EDIT: **Line** Circle Arc Curve Fill Zone Symbol Group Delay Bound Quit

Symbols are user-defined graphics objects that are stored in a library. For example, in the graphics shown in Fig. 3, the triangular objects were kept in the library as "inverters" and the objects with the semicircular ends were kept in a library as "Nand2 gates."

Graphics are added to a picture by selecting a graphic objects. When an object is selected, new commandlines appropriate to that object are displayed. For example, when Line is selected, the following commandline is displayed:

LINE: **Add** Delete Relocate Move Rotate Extend Zoom Undo Color Background Index Pause Default Line Circle Arc Curve Fill Zone >>

Graphics objects are created by placing points on the screen using the Add command. For example, if Line has been selected and points are added, a sequence of lines will be drawn through the points.

Video When a video frame is edited or created, the screen is cleared and the Video menu and commandline (see Fig. 4) will be displayed.

The Video Control Frame contains the data necessary to specify how the video is to be presented to a student. To edit an item in the Video Control Frame, move the cursor next to the item and choose the command Edit. For example, to edit the Start Frame, move the cursor next to it and choose the command

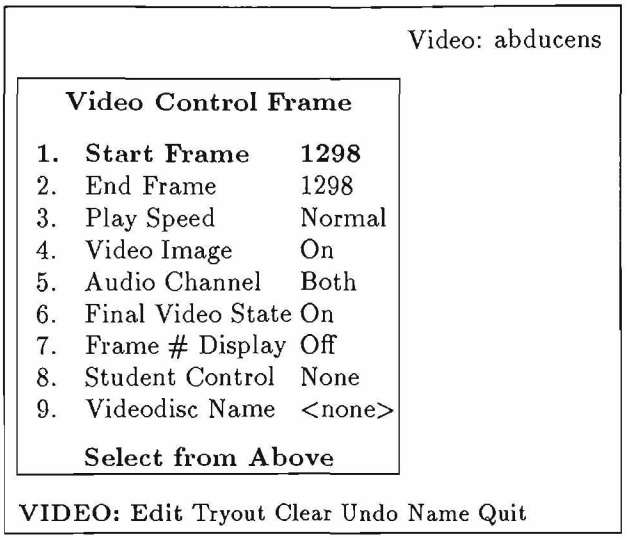


Figure 4: Video Control Frame

Edit. The Video commandlines will be displayed.

Halt_On Halt_Off Next Slow Normal Fast Browse Jump Direction Display Audio Keep Enter >>

The commands may be used to find the desired video material. If the frame number is already known, the Enter command may be selected to enter it.

3.3 Compilation

Authors use Fromtext to compile CBMS textfiles to the forms that can be used by the CBMS games. Fromtext is started by entering the word "fromtext" followed by the name of the CBMS text file. For example, to compile the CBMS textfile "cranial.txt"

```
fromtext cranial.txt
```

is entered in response to the operating system prompt. While Fromtext compiles a text file, it prints on the screen error messages and data that explain what it is doing. Three flags may be used with Fromtext: the flag 'q' turns off the screen display; the flag 'd' prints debugging information on the screen; and the flag 'o' specifies that any preceding file with the same name is to be overwritten. For example, to compile the CBMS textfile "cranial.txt" and to display debugging messages

```
fromtext -d cranial.txt
```

is entered in response to the operating system prompt.

Error messages from Fromtext are placed in a file with the same name as the source file; the extension is ".ERR." For example, the file "cranial.ERR" will contain the error messages generated when Fromtext compiles the file "cranial.txt." CBMS files should not be used with the games if any errors other than those associated with missing sequence files are detected.

The program Totext may be used to convert CBMS databases from the form used by the games to a text representation. Because Fromtext discards comments no comments will be present in the text file generated by Totext.

3.4 Testing

Fromtext detects the syntactic errors in CBMS textfiles. These errors must be corrected before testing for errors is begun.[14] The remaining errors can be found by using the games and Browser. Browser may be used with the flag 'b' to detect placement problems and poorly worded statements. For example, to detect problems with the database "cranial"

```
browser -b cranial
```

is entered in response to the operating system prompt. The flag 'b' permits examination of the db_other and db_relation categories. Because Fromtext distinguishes between nodes by the spelling of their names, a common problem is the accidental misplacement of an item because of a spelling error. A version of the item with one spelling will appear where it was anticipated, the other version will be placed in the db_other category "Not placed." Careful examination of the "Not placed" category will disclose these problems.

Other problems and errors are detected by playing the games. Common problems are not providing anticipated answers and misspelling item names.

4 Revision

Once a CBMS database is running and appears to be error free, it should be reviewed. This review supplements the initial reviews of the subject matter and addresses three issues:

- Matched_with statements
- Assigning importance
- Item exclusion

4.1 Matched_with Statements

The use of anticipated answers specified with MATCHED_WITH statements permits some latitude in student answers. However, the latitude is subject-matter specific; the range of an acceptable anticipated answer needs to be specified by experts in the field. Anticipated answers can be evaluated by having experts play the games and enter answers. Where their answers are not matched by an anticipated answer, the anticipated answer must be modified.[15] The problem of specifying an acceptable anticipated answer is similar to the problem of recognizing computer commands[7] with one major difference. While the objective in recognizing computer commands is to recognize the commands the user enters, the objective in recognizing anticipated answers is to accept only those answers that are acceptable to experts in the field. For example, "cranial nerve vii" is considered acceptable by neuroanatomists, but "cranial nerve 7" is not acceptable.

[14] Error messages resulting from missing sequence files can safely be ignored.

[15] Sometimes the questions are not sufficiently precise. Solution of this problem generally requires the addition of different relations and new sentence templates.

4.2 Assigning Importance

The Importance of items and statements in a database determines the minimum number of sessions in which students will have to answer questions involving the item or statement. One way to assign Importances is to divide the items and statements into three groups: critical, important, and needed but not important. The Importances can then be computed by determining the maximum number of separate sessions in which students can reasonably be expected to use the database; this number times 2 can be assigned to the critical items and statements. Then some fraction of this number can be assigned to the important and needed but not important items and statements. For example, 3/4 of the critical importance might be assigned to important items and statements and 1/2 of the critical importance might be assigned to needed but not important items.

4.3 Item Exclusion

In the revision process databases may become larger. Experts often are convinced that all items and statements that seem important to them must be added to a database. The consequence of these additions is a database that is either difficult or impossible for students to master in the allocated time. At this point, items and statements must be removed. There are two ways that this can be done. First, any items or statements that were not marked as at least needed can be removed. Second, entire categories that are not considered important or critical can be removed.

5 Summary

The Computer-Based Memorization System represents a method for using associative (semantic) nets, already widely used in Artificial Intelligence, for Computer-Aided Instruction. Three additions had to be made to associative networks to make them useful for education:

- Importance values had to be associated with items and statements to permit authors to emphasize features and to support files containing models of students' knowledge
- Pictures had to be associated with items
- Groups of anticipated answers had to be associated with items to allow a range of acceptable student answers

With these changes, CBMS appears to be an effective tool for the mastery of factual knowledge.

The Computer-Based Memorization System is one part of the Computer-Based Educational Software System (CBESS). Other parts of CBESS address vocabulary acquisition, equipment problem solving practice, instructional management, and script-based CAI. CBESS may be licensed.[16]

[16] Darbick Instructional Software Systems, P.O. Box 81157, Salt Lake City, Utah, 84108, licenses CBESS in

5.1 Acknowledgements

Drs. Henry Halff of Halff Associates and Douglas Wetzel of the Navy Personnel Research and Development Center evaluated the games and made many helpful suggestions. Richard B. Paulsen wrote the current versions of the games and Bradley N. Davis designed and implemented the libraries used by the CBMS programs.

The development of the CBMS programs was supported in part by the Navy Personnel Research and Development Center through Contract N00244-83-C-1759.

References

[1] J.R. Anderson. *The Architecture of Cognition.* Harvard University Press, Cambridge, MA, 1983.

[2] R.C. Brandt. *Sequence Editor.* Computer Science Department, University of Utah, Salt Lake City, Utah 84112, April 1987.

[3] R.C. Brandt, L. Gay, B. Othmer, and H. Halff. *Computer-Based Memorization System Author and Instructor Manual.* Computer Science Department, University of Utah, Salt Lake City, Utah 84112, April 1987.

[4] R.C. Brandt and R.B. Paulsen. *Computer-Based Memorization System Student Manual.* Darbick Instructional Software Systems, P.O. Box 81157, Salt Lake City, Utah 84108, July 1987.

[5] A.M. Collins and M.R. Quillian. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8:240–247, 1969.

[6] A.M. Crawford and J.D. Hollan. *Development of a Computer-based Tactical Training System.* Special 82-, Navy Personnel Research and Development Center, July 1982. Draft Version.

[7] S.T. Dumas, G. Furnas, L.M. Gomez, and T.K. Landauer. The vocabulary problem in human system communication. *Communications of the ACM*, 30(11):964–971, November 1987.

[8] N.V. Findler. *Associative Networks.* Academic Press, New York, New York, 1979.

[9] J. Grant and P. Marsden. The structure of memorized knowledge in students and clinicians: an explanation for diagnostic expertise. *Medical Education*, 21:92–98, 1987.

[10] T. McCandless. *Computer-Based Tactical Memorization System.* Technical Note 81-8, Navy Personnel Research and Development Center, March 1981.

[11] T.O. Nelson. Repetition and depth of processing. *Journal of Verbal Learning*

both source and executable form for use under UNIX and MS-DOS operating system. UNIX is a registered trademark of AT&T Bell Laboratories. and MS-DOS is a registered trademark of Microsoft, Inc.

and Verbal Behavior, 16(2):151–171, 1977.

[12] D.E. Rumelhart and D.A. Norman. Active semantic networks as a model of human memory. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, pages 450–457, 1973.