[*Proceedings of the National Symposium on Machine Translation*, UCLA February 1960]

Session 10: PROGRAMMING

MIMIC[1]: A TRANSLATOR FOR ENGLISH CODING

Hugh Kelly
The RAND Corporation

Summary

This paper describes an automatic coding system developed at The RAND Corporation to provide rapid implementation, testing, and modification of computer routines for linguistic research. A Translator analyzes and translates English-like statements into a pseudo-code program. An Interpreter subsequently executes the program. The system is being used to program rules for insertion and inflection in a Russian-English translation routine.

MIMIC is an automatic coding system developed[2] at The RAND Corporation and programmed for the IBM 704 computer. It enables the user to write computer routines for linguistic research as a series of English-like sentences. Each sentence is analyzed and translated into a compact pseudo-code by the MIMIC Translator. A second routine, the MIMIC Interpreter, subsequently uses this numeric code to select and execute the appropriate sequence of machine-language subroutines, thereby performing the action specified in the original English statement.

The system was developed to implement, test, and modify linguistic algorithms with a minimum of time and effort. It frees the programmer from a multitude of time-consuming details by allowing him to write programs at a language level considerably higher than that of the computer. The user chooses and defines his own vocabulary; thereafter he is free to use this terminology without adverting to the details of the variables about which he writes.

MIMIC was designed specifically to program the rules for insertion and inflection of English equivalents in the RAND Russian-English translation routine. The tree-like codes that embodied these algorithms were subject to frequent modification in order to reflect changes in the grammar code, refinements in our theory of structure,

---

[1] MIMIC is not an acronym. The name is intended to reflect the fact that the system translates an "imitation" of English.

[2] I wish to thank my colleagues M.I. Bernstein, I.D. Greenwald, D.G.Hays, C.H. Smith, and T.W. Ziehe for the helpful suggestions which they contributed to the development of the system.

and improvements in the algorithms themselves. This section of the routine performs a relatively simple task; for each word in the sentence, a series of conditional tests determines a branching path through the code. Each path terminates with the selection of the insertions, deletions, inflections, or substitutions that are needed for this word; four terminal subroutines perform the indicated operations on the English equivalent. The latest revision of insertion and inflection rules is currently being programmed in MIMIC.

The system allows three kinds of statements. The programmer uses imperative and conditional statements in writing his program; these are compiled into a pseudo-code routine by the Translator. He uses definitional statements to define new terminology; these statements are not compiled into pseudo-code but cause new words and their meanings to be entered in the Translator's dictionary.

Some imperative statements specify simple control commands, such as "Go to location-x" or "Exit interpreter". Other imperative statements specify operations at a higher level, such as "Insert 'THE' ahead of this-word, then go-to k-1". MIMIC currently contains 8 such operations that deal with inflection and insertion. New operations are added to the language (or dictionary of MIMIC) by definitional statements; subroutines to perform these actions must be provided, written either in MIMIC or in machine-language.

A conditional statement specifies a condition to be tested and the action to be performed if the test is successful; control goes to the following statement if the condition is not met. For example:

```
IF POSITION-1 OF THE GRAMMAR-CODE OF THIS-WORD = 2,
GO-TO K-1.
```

An imperative statement may be joined:

```
IF POSITION-1 OF THE GRAMMAR-CODE.....= 2, INSERT
(THE) BEFORE THIS-WORD, THEN GO-TO K-1.
```

Conditional N-way switch statements are convenient for branching:

```
SWITCH ON POSITION-1 OF THE GRAMMAR-CODE OF THIS-WORD,
OR GO-TO X-1.
IF = 2, GO-TO K-1.
IF = 4, ADD(-ING) AND GO-TO P-5.
IF = 6, INSERT (TO) AND EXIT.
```

The Translator is equipped with a dictionary of source-language words. Every dictionary entry contains the following information:

1. A source-language word of 12 alphanumeric characters or less
2. A classification-code symbol for the source-language word
3. A target-language equivalent, or meaning

The classification-code symbol for a word indicates what function it performs in the artificial source language. There are 10 classes of functions to specify: (1) a test, (2) an imperative command, (3) an item, (4) a mask, (5) an index, (6) a relation, (7) a single-character value, (8) a multiple-character value, (9) an operand, (10) a location. A word may belong to only one class, i. e. , have only one primary function. But as we will see later, it may have secondary functions in certain contexts. Similarly, a word has only one primary equivalent.

This lack of ambiguity in function and meaning in the source language would seem desirable; but, in fact, MIMIC's inability to assign several primary meanings to a word for later resolution in the context of a particular statement is restrictive. An example will illustrate the point.

Consider the word "genitive", an English adjective whose meaning is unambiguously the same in the phrases "a genitive noun" and "a genitive adjective". In both, it denotes the case of a word.

But the word "genitive" should have an ambiguous meaning in the MIMIC dictionary; that is, it should have two distinct equivalents, and selection between them should be determined by the context of the particular statement in which "genitive" appears. The two meanings are needed because the RAND grammar code represents the genitive case of a noun by one symbol and the genitive case of an adjective by a different symbol. The ability to make contextually resolvable, ambiguous definitions would allow us to specify the two tests for genitive case with the same word:

```
IF CASE-OF-NOUN = GENITIVE GO-TO K-1, and IF CASE-OF-
ADJECTIVE = GENITIVE, ...
```

where "genitive" would be defined by:

```
D-1 DEFINE GENITIVE = X FOR CASE-OF-NOUN, AND = Y FOR
CASE-OF-ADJECTIVE.
```

Inability to handle ambiguous definitions makes us use somewhat inelegant expressions to accomplish the same tests, viz:

```
IF CASE-OF-NOUN = GENITIVE-NOUN, GO-TO K-1 and IF CASE-
OF-ADJECTIVE = GENITIVE-ADJECTIVE, ...
```

where "genitive-noun" is set equal to x and "genitive-adjective" is equated to y .

Let us consider a simple algorithm to see how it can be expressed in MIMIC. Assume that we wish to test the part-of-speech classification of each word in a sentence, determining whether the word is a noun. If it is, we wish to branch to another part of our program to insert the article "the" if the case is nominative; if it is not a noun, we will increase the word-counter by 1 and repeat the test on the next word in the sentence until all have been tested. Assuming that a noun is identified by having "2" in the first (left-most) position of its grammar code, and that nominative case is represented by "7" in the third position, we can write the following MIMIC Statements:

NAME	STATEMENT
J-1	IF POSITION-1 OF THE GRAMMAR-CODE OF THIS-WORD = 2, GO-TO K-1.
M-1	IF WORD-TALLY ≠ END-OF-SENTENCE, INCREASE WORD-TALLY BY 1, GO-TO J-1. OTHERWISE EXIT.
K-1	IF POSITION-3 OF GRAMMAR CODE OF THIS-WORD = 7, INSERT 'THE', GO-TO M-1. GO-TO M-1 ANYHOW.

Figure 1 MIMIC PROGRAM FOR INSERTING ARTICLE

NOTE: A hyphenated phrase is considered one word. Words may be of any length for readability with the restrictions that (1) the total statement not exceed 72 characters, and (2) words must differ in the first 12 letters to be unique. The above violate restriction 1 for clarity of exposition.

To translate J-1, the MIMIC Translator scans the statement, isolates a word, and attempts to find the word in its dictionary. The dictionary entry for a word contains the meaning for the word and also a classification symbol telling what type of word it is, i. e. , what function it can perform in a sentence. A simple conditional statement must contain the types of words shown in Figure 2.

	Type of Word Needed	Satisfied in J-1 by:
1.	A conditional word	IF
2.	An item-specifier	GRAMMAR - CODE
3.	A mask for the item	POSITION-1
4.	An index for the item	THIS-WORD
5.	A relation symbol	=
6.	A value or values that satisfy the test	2
7.	The location of the next statement to execute if the test is successful	K-1

Figure 2 CONTENT OF A MIMIC STATEMENT

The underscored words in Figure 2 are not part of the basic MIMIC system, so we must enter them in the dictionary with definitional statements, specifying what function we want them to perform and assigning an equivalent. We do so with the following statements:

D-2 DEFINE GRAMMAR-CODE AS ITEM-07.

D-3 DEFINE POSITION-1 AS MASK-01.

D-4 DEFINE THIS-WORD AS INDEX-04.

The first statement enters the word "grammar-code" in the dictionary, classes it as an item-specifying word, and gives it the meaning "07". Whenever "grammar-code" is encountered in a statement now, its pseudo-code translation will be "07". If the Interpreter routine is to execute this pseudo-code properly, the 7th name on its list of item-fetching subroutines must refer to the subroutine that fetches grammar codes. Terms like "item-07", "mask-01", and "index-04" are provided as system symbols that enable the programmer, when first building a vocabulary, to define his basic terminology in terms of their function in MIMIC. Thereafter, he adds new words by defining them in terms of this basic terminology. Thus, given definitions D-2 and D-3 above, we will later write:

```
D-6  DEFINE PART-OF-SPEECH AS POSITION-1 OF THE
GRAMMAR-CODE.
```

Since statement J-1 tests whether a word is a noun, we can clarify its intent by rewriting it as:

```
J-2  IF POSITION-1 OF THE GRAMMAR-CODE OF THIS-WORD =
NOUN, GO-TO K-1.
```

Assuming that the MIMIC dictionary does not yet contain the word "noun", we enter it with:

```
D-5  DEFINE NOUN = 2 .
```

The word "noun" can now be used interchangeably with "2" in specifying values that satisfy a condition. Furthermore, we have committed ourselves to using the word "noun" for no other reason than to specify a value. This restriction will be modified by a more useful definition of the word "noun" as the discussion develops.

Although statement J-2 is more descriptive than J-1, we note that the words "position-1" and "grammar-code" remain as concessions to the computer's demand for explicit detail. It is reasonable to replace the words "position-1" of the "grammar-code" with the single, more descriptive term "part-of-speech" since this is what position-1 of the grammar-code contains. The replacement changes our statement to:

```
J-3  IF PART-OF-SPEECH OF THIS-WORD = NOUN, GO-TO K-1.
```

Before statement J-3 can be correctly translated, the MIMIC dictionary must be provided with a definition of "part-of-speech", as follows:

```
D-6 DEFINE PART-OF-SPEECH AS POSITION-1 OF THE
GRAMMAR-CODE.
```

Let us postpone discussion of this definition for a moment in order to compare statements J-1 and J-3.

```
J-1    IF POSITION-1 OF THE GRAMMAR-CODE OF THIS-WORD =
2, GO-TO K-1.
J-3    IF PART-OF-SPEECH OF THIS-WORD = NOUN, GO-TO K-1.
```

J-1 specifies the test in computer-oriented terminology, whereas J-3 specified the same test in terms that make the meaning of the test clear at a glance. J-1 specifies the object of the test in terms of its data format, "position-1 of the grammar-code"; J-3 specifies the object of the test in terms of its significance to the programmer, "part-of-speech", without reference to the details of data format.

J-3 contains two symbols, "part-of-speech" and "noun", which the user himself was forced to define or do without. Hopefully, because he defined them, he chose them to be appropriate to his problem, he understands their meanings, and he will find them easy to recall for future use.

Returning to the definitional statement D-6, notice that it is a natural and logical definition in the context of the problem; part-of-speech information can only be found by examining the first character of the grammar code. D-6, in effect, equates the term "part-of-speech" with a particular character position within the machine word "grammar-code". This is merely an extension of the practice of symbolic naming that is common in symbolic assembly programs. Since the machine word "grammar-code" contains distinct, nameable pieces of information in different character positions, these pieces can each be given a single distinct name in terms of their character positions. It is easier to talk about "case", "number", and "gender", than to refer to the same information as "position-3 of the grammar-code", position-4 of the grammar-code", etc.

The definitional statement above creates a slightly different type of dictionary entry from that of our previous definitions. ''Part-of-speech" is still given only one primary classification and meaning. Like "grammar-code", it is classed as an item-specifying symbol and has the same primary meaning as "grammar-code". In addition, however, it is allowed to imply the meaning assigned to the masking symbol "position-1" if the context of the particular statement warrants it. That is, whenever the word "part-of-speech" occurs in a statement, it always and unconditionally specifies item "grammar-code";

it will also specify the masking symbol "position-1" provided that the statement does not contain an explicit masking symbol. In other words, the relationship established by the definition D-6 between the term "part-of-speech" and "position-1" is conditional; "part-of-speech" implies "position-1" on the condition that no other masking symbol explicitly occurs in the same statement. But the relationship which D-6 establishes between the terms "part-of-speech" and "grammar-code" is unconditional; "part-of-speech" always specifies the item-symbol "grammar-code". The conditional nature of implied definitions allows the programmer to override them, when this is useful, by putting explicit symbols of the same class in his statement. This ability to override is not useful in this example, but let us now redefine "part-of-speech" as follows:

```
D-7  DEFINE PART-OF-SPEECH AS POSITION-1 OF GRAMMAR-
CODE OF THIS-WORD.
```

"Part-of-speech" now contains two implied symbols; namely, a masking symbol "position-1", and an index symbol "this-word". "This-word" is a variable that can be evaluated only when the pseudo-code is being executed by the Interpreter. At that time it successively assumes the values of a current-word counter that steps through each word of the sentence being translated. Since so many of the statements specify tests on the current-word, it is convenient to avoid writing "this-word" each time; D-7 allows us to do so, since "part-of-speech" now implies that we are referring to the part-of-speech of the current word. Hence we can shorten J-3 by:

```
J-4   IF PART-OF-SPEECH  =  NOUN,  GO-TO  K-1.
```

When, however, we wish to test a word in the sentence that is structurally related to the current word, we can override the implied index symbol by specifying one explicitly in our statement:

```
J-5  IF PART-OF-SPEECH OF GOVERNOR  = NOUN,  GO-TO K-1.
```

Here "governor" is defined by:

```
     D-8     DEFINE GOVERNOR AS INDEX-05.
```

Again, D-8 merely enters the term "governor" in the Translator's dictionary, classifying it as an index symbol and assigning it the meaning "05"; if the Interpreter is to execute the statement correctly, the 5th name on its list of index-setting subroutines must refer to a subroutine that will point to the syntactic governor of the current word. Since a word has only one governor, J-5 results in only one test. A word may have several dependents, however, either preceding or following it in the sentence. We may wish to write:

```
J-6  IF PART-OF-SPEECH OF PRECEDING-DEPENDENT  = NOUN,
GO-TO  K-1.
```

Here "preceding-dependent" is defined by:

```
D-9   DEFINE PRECEDING-DEPENDENT AS INDEX-06.
```

J-6 may result in several tests, since all preceding dependents of the current word must be examined until either a noun is located, or all preceding dependents of the current word have failed the test. The Interpreter executes J-6 iteratively, using, as different values for the term "preceding-dependent", those values which the 6th index-setting routine supplies. The ability to write statements that specify both a structural relationship and grammatic qualities as conditions is expected to be especially useful in writing programs for post-translation analysis of corrected text.

Returning to the concept of implication, consider the two sentences

> "The color of his eyes was blue."
> and
> "His eyes were blue."

The first sentence explicitly names which feature of the eyes is being described; namely, color. The second sentence implies the same information in the word "blue", and is somewhat less stilted. The MIMIC statements we have been considering resemble the first sentence; the feature being tested has been stated explicitly. The ability to imply this information would not only make the statements more natural, but would save repetitious writing. Let us change the definition of the word "noun" from D-2 to;

```
D-10   DEFINE NOUN AS PART-OF-SPEECH  =  2 .
D-11   DEFINE NOUN AS POSITION-1 OF GRAMMAR-CODE  OF
THIS-WORD  =  2.
```

The dictionary entry for the word "noun" is now carrying the burden of detail, and the programmer can write such cryptic statements as:

```
J-7   IF GOVERNOR  = NOUN, GO-TO K-1.   (EQUIVALENT TO J-5)
J-8   IF PRECEDING-DEPENDENT  =  NOUN, GO-TO  K-1.
(EQUIVALENT TO J-6. )
```

Unfortunately, the present format of the dictionary entry does not provide space to define a value and, at the same time, to imply an item, a mask, and an index. It does provide space for an implicit item and mask, however. Implicit masking symbols have already been incorporated; implicit item-symbols will be added.

Between 6 and 7 man-months have been spent on the design, programming, and check-out of the MIMIC system; and it is currently being used to construct English sentences in the Russian-English translation routine. If, as expected, the system is to be used for analysis of postedited text, certain action subroutines for tallying, sorting, listing, and similar operations, will probably be added to the Interpreter in machine language. The ability to write action subroutines in MIMIC statements has recently been added but needs refinement before it will be of real value.

Clearly, there has been an attempt to make the system easy to use, in the hope that non-programmers could master it with a little training. It is too early to predict whether this generally will be the case, although one such person has written several routines correctly in the system.

Interpretive systems are somewhat unappealing, and for some problems even unacceptable, because by nature they are less efficient in machine time. We have no measure of MIMIC's efficiency at present, but we expect it to be tolerable; the routines for which it will be used are characterized by having many branches, only one of which is selected for a given word. The machine-language codes for inflection and insertion consumed less than 10 percent of the total internal

machine time for the Russian-English translation, but a much higher percentage of total available programming time. At this stage of research, we are inclined to speculate that the ability to mechanize five algorithms less efficiently will produce linguistic progress faster than the ability to mechanize one of them optimally.