[*Automatic Translation of Languages*: NATO Summer School…July 1962]

AUTOMATIC TRANSLATION OF LANGUAGES*

A. F. R. BROWN
Georgetown University, Washington, D.C.

1. MACHINE TRANSLATION AS A TASK—AVAILABLE TOOLS AND GENERAL METHODS

THERE are various possible ways of thinking about mechanical translation of languages; one of them is as a task which deserves to be attempted. We know that a perfect translation cannot be achieved; we also know that a crude word-for-word translation can be carried out by machine. The interesting question is: How good a translation can we make by computer in the foreseeable future?

There is no way to answer this question without trying. *A priori* answers will range from 'a translation which will eliminate most human translation work' to 'no translation which could under any circumstances be worth having'. No doubt the truth lies in the middle, but in dealing with something as intractable as natural language there is really no way of predicting what can be achieved in a few years of intelligent work without doing the work. So I intend to discuss machine translation in the context of a hypothetical task, that of working for two or three years on a pair of languages, with the expectation that the work will culminate in putting some kind of translation system on a computer; not in order to prove a thesis about the potential of automatic translation, but in order to see what can be achieved; what unexpected problems come to light; what expected problems turn out to be manageable; and what problems are brought closer to solution by the help of mechanical data-processing methods.

It is worth beginning with a discussion of machines. Everyone knows that electronic computers can store large amounts of information and can make decisions very rapidly. The question is how their powers compare with the requirements of an automatic translation system. I shall consider the IBM 7090 as a typical machine of the sort that one would consider first for machine translation. It is not necessarily the most efficient in its class, but it is not, at any rate, much below the leader in efficiency. The class of computers including the 7090 is not quite the most powerful in existence, but any computer that is markedly more powerful than the 7090 is not easily available today. On the other hand, any machine much smaller than the 7090 class will necessarily be less efficient for most purposes, including machine translation. By 'less efficient' I do not mean that a worse translation would necessarily be produced by a smaller computer. In theory, any automatic translation algorithm which the wit of man can devise could be programmed for any computer with magnetic tapes. But as a general rule, a machine which is twice as small will handle any algorithm more than twice as slowly; so it will be less efficient in terms of running cost.

The first difficulty that usually comes to mind is the dictionary. Making a dictionary is very laborious for human beings, but storing it and using it are no problem for a 7090. Let us imagine a dictionary with 100,000 entries. This looks like a very generous estimate of

* Presented at the NATO Advanced Study Institute on Automatic Translation of Languages: Venice, 15-31 July, 1962.

the number that might be necessary for translating one language to another. Now imagine that each entry contains, on an average, 100 characters. This is also a rather generous estimate, allowing 15 characters for the word in the input language, 15 characters for its equivalent in the output language, and 70 characters for whatever other information will be needed. Seventy characters is equivalent to 105 ordinary numerical digits, or 420 binary digits; i.e. this amount of space can furnish the answers to as many as 420 'yes/no' questions. For the average dictionary entry, this is generous. So the size of the imaginary dictionary will be about 10,000,000 characters. This is comfortably within the capacity of one reel of magnetic tape for the 7090. Since a 7090 normally has room for at least ten magnetic tapes to be attached to it at once, it is clear that the mere size of the dictionary will not present a storage problem.

The next question that naturally arises is: Given such a dictionary, how can it be consulted with reasonable speed? Once again, the fact is that quite conventional methods will suffice. One magnetic tape can be read by the 7090 at the rate of 60,000 characters/sec; most 7090 computers actually have two channels for magnetic tape reading and writing, which can be used simultaneously. This means that they can read in information at the rate of 120,000 characters/sec. (It is possible for a 7090 to have as many as eight channels, through which it could read almost 480,000 characters/sec, but in practice two channels is the usual equipment.)

At 120,000 characters/sec, the imaginary dictionary of 10,000,000 characters could be read by the computer in 80 or 90 sec. The way to minimize the average time needed to look up a word in the dictionary is to make sure that every time 80 or 90 sec are spent in reading through the dictionary, as many words as possible are looked up during the reading. It is not difficult to look up 2000 words during one reading of the dictionary, thus using an average of 40 msec to look up each word. With a more complicated program, the size of the batch can be increased from 2000 words to 4000 or even 10,000, so that if necessary the average dictionary time per word could be reduced to 8 or 9 msec.

The way in which a single reading of the dictionary serves to look up several thousand words is very simple in principle. If a human being had to look up two thousand different words in a large dictionary, he might write each word on one card, sort the cards into alphabetical order, and then work through the pile of cards during a single reading of the dictionary from *A* to *Z*. For a computer, the sorting of the words into alphabetical order is a simple procedure, but the time required for the sort increases faster than the size of the group of words to be sorted; thus there is some optimum number of words to look up during a single reading of the machine dictionary. This optimum number will depend on several factors; 2000 words in a batch is certainly less than the optimum number.

It is possible to imagine quite different strategies for dictionary look-up, and storage devices other than magnetic tape may be invented for dictionaries. But in the present stage of development I think it is worth emphasizing that with a standard, general-purpose computer of the present day, using quite a simple program, one can consult a very large dictionary at the rate of at least 25 words/sec, or, to put it into a different light, 90,000 words/h. Thus the problem of dictionary size is not a critical one in machine translation. The real problems lie elsewhere, and research aimed at increasing the speed of dictionary-handling cannot do much to help the development of machine translation. Such research is worth doing, of course, but it can only improve the efficiency of a machine translation system; it cannot improve the quality of the translation, or even change a proposed system from unfeasible to feasible.

Of course a dictionary is not enough. It can only furnish the raw material which must somehow be converted into a translation, or at least the best possible approximation to a translation. The computer is called on to make a great many decisions and changes in the raw material, between the time it comes from the dictionary and the time the finished product is written out. This processing actually takes longer than the dictionary look-up.

A 7090 computer has a memory access time of a little less than 2.5 μsec, and an ordinary instruction time of somewhat less than 5 μsec. That is, it can carry out about 200,000 instructions/sec. If one compares one of these 7090 instructions to an atom, then a simple linguistic decision will be comparable to a molecule of some protein substance. No exact numerical comparison can be given; but suppose the computer has to count the number of adjectives in a sentence, and that the number is three. This counting might require something between a hundred and a thousand 7090 instructions to be executed, and this would take from 0.5 to 5 msec. It is easy to believe that a thousand decisions like this would have to be made during the processing of one sentence of average length, consuming between 0.5 and 5 sec. In my own experience, the actual time will be closer to the lower limit—say 1 sec, for simplicity in discussion.

Then for a sentence of ten words, with a dictionary system like the one I have described, 400msec would be consumed in looking up the words in the dictionary, and 1000msec in turning what was taken from the dictionary into what is written out as the translation. These figures are all so approximate that they cannot prove anything in detail. But they suggest, I hope convincingly, that a huge dictionary does not present any great problem of speed; that the logical decisions involved in translation take longer than the mere consultation of the dictionary, and that an over-all speed of 25,000 words/h is easy to achieve, as far as machine speed is concerned. This rate of production would not be commercially profitable, but it would certainly be adequate for purposes of research and development. In other words, the difference between 10,000 and 100,000 words/hr might well be the difference between loss and profit if machine translation were attempted commercially, but it would not be the difference between feasibility and unfeasibility in the work of developing a system of machine translation.

I have gone into some detail about speeds because I am convinced that speed, at present, is no problem, and one's primary attention should be fixed on the questions of finding methods for translation by rigorous, non-intuitive procedures, and of facilitating the human work of communicating these procedures to the machine, interpreting the results given back by the machine, and correcting the procedures.

Most machine translation research projects show an obvious division into a linguistic section and a programming section. This reflects not only a certain natural division of skills between linguists and programmers, but also the division between information about languages which ought to be valid without reference to any machine, and techniques of giving the information to a particular machine for a particular purpose. One would suppose that the division between linguists and programmers is unfortunate, even if necessary. In principle it is unfortunate, because communication between the linguists and the programmers is never easy. But in practice, the division is always broken down to some extent, where it ought to be rigorously maintained.

In the early stages of producing some kind of program and bringing it to a machine for testing, a procedure is usually adopted by which the linguists write the dictionary, and the statements of procedure, in a form which will be as easy for the programmers to handle as possible. Three evils result from this. First, the programmers lose their freedom. Too close a

liaison is established between linguists who understand a very little about programming, and programmers who understand a very little about languages; and the basis of their liaison is the first attempt at a system, which will need to be superseded completely in a year. Second, the programmers are made lazy because their job looks deceptively easy at the beginning. Third, linguists get the habit of talking in numbers instead of words—for some people, it is more enjoyable to say "This item has code 2 in position 36" than to say "This item is a verb". Some linguists may even be seduced into playing with numerical methods for solving problems that are not at all numerical.

One of the greatest difficulties in getting linguistic statements into a machine is the need for rigor. When a competent linguist begins to make statements for machine translation, he may not realize that these statements have to represent procedures, not merely passive descriptions. Once he begins making procedural statements, most of them will be, strictly speaking, unprogrammable. They will leave too many possibilities unaccounted for, and will probably contain endless loops. It is up to the programmers to add rigor to the linguistic statements, and to get the linguists' informed consent to this. At first the linguists are reluctant, but later they are too willing to leave the burden of rigor to the programmers, especially if there is a satisfactory liaison between the two groups. Yet the best results in machine translation research cannot be obtained until the linguists can devise procedures that are not only ingenious, but also well-organized.

So I believe that from the very beginning of a research effort, the linguists should try to formalize their statements in some kind of coding language which forces them to be rigorous and procedural, but is not much affected by considerations of how it will ultimately be implemented by a computer. Such a coding language should be developed and improved to suit the linguists, not the programmers. The programmers will have the job of making the computer accept and implement statements in the coding language. This is a formidable task, but in fact it can be a satisfactory one for the programmers because it is not endless, and leaves them their proper freedom. The machine translation system, as expressed in the special coding language, will never be completed, but the computer programs for operating on the special coding language can be completed; and they can be altered or moved to a different computer without disturbing the linguists.

The advantage to the linguists will be that they, or at least some of them, will become accustomed to making their statements in a rigorous way; they will not be dependent on the programmers for having the T's crossed and the I's dotted.

Because I believe that the purely mechanical problems of automatic translation should not concern linguists once they have begun work on a pair of languages, I shall say no more about computers as such, or about bits, machine words, memory sizes, and so on. I shall speak in terms of an imaginary system of coding dictionary entries and linguistic operations, and assume that statements in this coding language could be accepted and implemented by a suitable program on any computer. The only way in which this coding language will be conditioned by electronic realities is that it will use only a standard set of IBM characters: the twenty-six letters, the ten digits, the blank, and the extra symbols = + - $ * ( ) , / and .

2. ARABIC-TO-ENGLISH TRANSLATION AS A HYPOTHETICAL PROBLEM—MORPHOLOGY AND DICTIONARY CODING

I have chosen Arabic-to-English machine translation as a hypothetical task, in order to illustrate some simple general methods of attack on MT problems. A European language would have had the advantage that less preliminary explanation would be necessary; on the

other hand you all know too much about European languages to come to any of them with a fresh mind. I do not want to describe an input language with the proper linguistic rigor; this would take too long, and in fact such a description is not the proper starting point for machine translation. On the other hand, I do not want to describe a European input language summarily, and leave all those who know the language dissatisfied from the very beginning. So I chose Arabic, which is not enormously different in its syntax from a European language in any case.

When I started writing this paper, I was trapped, indirectly, by exactly the error which I intended to combat most vigorously. The syntax of Arabic is simpler than that of any European language, and it has been well-described for centuries. This means that one could, in some sense, 'program the grammar' of Arabic quite easily—much more easily than for Russian. But in fact this does not help as much as I unconsciously expected, and you will notice a progressive loss of exhaustiveness as these lectures go on. The descriptive grammar of the input language has to be known, but programming it would be no help; we need a recognition procedure, and this is infinitely more difficult.

Printed Arabic represents, not quite completely, a language with twenty-eight consonants, three short vowels, three corresponding long vowels, and a possibility of doubling consonants. Normally, the short vowels are not represented in printing at all; the long vowels are represented by consonant signs; and the doubling of consonants is not represented. This should leave a remarkably simple system, though one which is sometimes a little hard to read. There are a few residual complexities, however. One of the consonants has a special symbol when it represents a feminine ending (this is a compromise between two pronunciations of the feminine ending). Another consonant (the *hamza,* or glottal stop) can be represented in several different ways, according to its surroundings. One of the long vowels can be written in two different ways at the end of a word, according to its etymology.

Now obviously the first problem of machine translation from Arabic to English is how to represent Arabic on punched cards, using only an IBM set of characters, without losing information available in the printed text, and without adding any information from the card-puncher's own knowledge of Arabic. The transliteration I am about to describe fits these conditions (with an exception noted for the letter I), but I shall not stop to justify it in detail. For the benefit of those who know Arabic, or who will be curious afterwards, I will merely sketch the correspondences between characters as printed in Arabic and the characters and digraphs to be keypunched.

A alif without hamza; or with hamza above, *if* at the beginning of a word.
AA alif with madda
I hamza with no supporter; or alif, waw, or ya with hamza *except* alif with hamza above it at the beginning of a word. (A very slight loss of information is accepted here for the sake of simplicity in exposition.)
B ba
T ta
X ta marbuta (the special form of 't' used as the feminine ending for nouns and adjectives).
T/ tha
J jim
H/ ha
K/ kha
D dal

D/ dhal
R rai
Z zain
S sin
S/ shin
S. ṣad
D. ḍad
T. ṭa
Z. ẓa
C ain
G ghain
F fa
Q qaf
K kaf
L lam
M mim
N nun
H ha
W waw
Y ya (with dots)
A/ ya, without dots, at the end of a word (ya without dots in the middle of a word occurs only as the supporter of hamza, and so is represented by 'I').

In printing, the three long vowels 'aa', 'ii', and 'uu' are represented by *alif* without *hamza, ya,* and *waw;* thus in the transcription I am using they will be represented by A, Y and W.

Arabic words represented in this transcription are going to appear unpronounceable. This is unfortunate, but ordinary printed Arabic is only pronounceable because a person who knows the language knows what unwritten short vowels to introduce between the consonants, and knows when Y and W represent consonants and when they represent long vowels. However, present-day machine translation research deals only with the printed word; so the unpronounceability is inconvenient but not significant.

I shall now give an outline of Arabic morphology as it would appear in this transcription. To reduce the size of the presentation, certain classes of forms which occur comparatively rarely in ordinary printed texts have been left out: second person forms, third person plural feminine forms, and all dual forms. These omissions eliminate certain kinds of homography, and therefore simplify certain problems illegitimately; but enough kinds of homography remain to illustrate the general problem. For those who have studied written Arabic, it is also worth noting that I am ignoring here the traditional ten or fifteen 'forms' in which an Arabic verbal root can appear. The 'forms' are important to a descriptive linguistic study of Arabic morphology, but from the point of view of machine translation they are bound to be treated as separate lexical items.

Nouns

An Arabic noun is either masculine or feminine; it displays one of three cases (nominative, accusative, or genitive); one of three numbers (singular, dual, or plural); and one of three 'states'—definite (defined by a preceding definite article), construct (defined by an immediately following noun in the genitive case), or indefinite (not defined in either way).

Here is the paradigm of a masculine noun, NJAR ('a carpenter'), leaving out the dual:

	Sing. Def. or Cstr.	*Sing.Indef.*	*Plur. Def. or Indef.*	*Plur.Cstr.*
Nominative	NJAR	NJAR	NJARWN	NJARW
Accusative	NJAR	NJARA	NJARYN	NJARY
Genitive	NJAR	NJAR	NJARYN	NJARY

This suggests how a dictionary entry and a paradigm might be coded, in a hypothetical coding system for the linguist:

```
NJAR = P1    NOUN+MASC   *CARPENTER $S
$  PARADIGM 1 (FOR MASCULINE TRIPTOTE NOUNS WITH SOUND
   PLURALS)
-        SING+(NOM/ACC/GEN+DEF/CSTR)/(NOM/GEN+INDEF)
A        SING+ACC+INDEF
WN     PLUR+NOM + DEF/INDEF
YN      PLUR+ACC/GEN+DEF/INDEF
W       PLUR+NOM +CSTR
Y        PLUR+ACC/GEN+CSTR
```

Unfortunately this is not quite the whole story. To any of the construct forms one may add a 'possessive suffix'; thus the suffix NA (meaning 'our') would produce the forms NJARNA, NJARWNA, and NJARYNA. While these suffixes are treated in the traditional grammar as separate words, they are written as suffixes, and so machine translation morphology must treat them as part of the noun inflection. The list of suffixes (excluding duals, second person forms, and the third person plural feminine form) is: Y ('my'), H ('his'), HA ('her'), NA ('our'), and HM ('their'). When the suffix Y is added to either of the construct forms NJARW and NJARY, the result is simply NJARY.

So we need a rather more complicated paradigm:

```
$    PARADIGM 1 (FOR MASCULINE TRIPTOTE NOUNS WITH SOUND
   PLURALS)
-            SING+(NOM/ACC/GEN+DEF/CSTR)/(NOM/GEN+INDEF)
Y           (PLUR+ACC/GEN+CSTR)/(SING/PLUR+NOM/ACC/GEN+MY)
H           SING+NOM/ACC/GEN+HIS
HA          SING+NOM/ACC/GEN+HER
NA          SING+NOM/ACC/GEN+OUR
HM          SING+NOM/ACC/GEN+THEIR
A            SING+ACC+INDEF
WN         PLUR+NOM+DEF/INDEF
YN          PLUR+ACC/GEN+DEF/INDEF
W           PLUR+NOM+CSTR
WH         PLUR+NOM +HIS
WHA      PLUR+NOM +HER
WNA      PLUR+NOM +OUR
WHM     PLUR+NOM+THEIR
YH         PLUR+ACC/GEN+HIS
YHA       PLUR+ACC/GEN+HER
YNA       PLUR+ACC/GEN+OUR
YHM      PLUR+ACC/GEN+THEIR
```

NJAR is actually untypical of Arabic nouns, in that it has a plural formed from the singular by adding an ending (WN W YN or Y). The vast majority of nouns form their plurals by modifying their stems, in ways which are only sometimes predictable. For instance, the plural of KTAB (book) is KTB; the plural of S/BAK (window) is S/BABYK. These plurals, traditionally called 'broken plurals', are inflected like singulars. So we can set up another paradigm:

```
$  PARADIGM 2 (FOR MASCULINE TRIPTOTE SINGULARS AND BROKEN
   PLURALS)
-       (NOM/ACC/GEN + DEF/CSTR)/(NOM/GEN + INDEF)
Y       NOM/ACC/GEN + MY
H       NOM/ACC/GEN + HIS
HA      NOM/ACC/GEN + HER
NA      NOM/ACC/GEN + OUR
HM      NOM/ACC/GEN + THEIR
A       ACC +INDEF
```

For KTAB and KTB we shall have two dictionary entries:

```
KTAB = P2   NOUN+MASC +SING   *BOOK $S
KTB = P2   NOUN + MASC + PLUR   *BOOK $S
```

Unfortunately, the left-hand end of a noun is not stable. The definite article, ordinarily spelt AL, is never written as a separate word; it is prefixed to a noun or adjective, which must be in the definite state (and therefore not followed by a possessive suffix). For NJAR and KTAB, the possible forms with the definite article are ALNJAR, ALNJARWN, ALNJARYN, ALKTAB, and ALKTB.

Furthermore, there are three prepositions, B L and K, which can precede any genitive noun form, and must be joined to it in writing. So forms like BNJAR, BALNJAR, BNJARYN, BALNJARYN, BNJARYNA, BKTBHM are possible. When the preposition L precedes the definite article, the combination is written LL, not LAL.

Finally, either the copula W or the conjunction F can be added at the beginning of any word. So we can say that the set of possible prefixes to nouns is AL WAL FAL B WB FB BAL WBAL FBAL L WL FL LL WLL FLL K WK FK KAL WKAL FKAL W and F.

To say the least, these prefixes pose a problem. I have already pointed out that in order to speed up the use of the dictionary, it is essential to have a large group of words from the input text sorted into alphabetical order. This is quite straightforward for a language like French or Italian, in which inflection takes place only at the end of a word. If a language were inflected only at the beginnings of words, one could reverse the strategy; or indeed one could reverse every word in the text and also reverse the words in the dictionary, and then follow the same strategy as for French.

But when both ends of a word may be inflected, it is difficult to see how to put it into alphabetical order with other words. For Arabic, the following solution will work. The program which reads the input text and stores the Arabic words must be directed to examine every word to see whether it begins with one of a list of allowable prefixes. If not, the word is handled just as it is; but if so, the word is handled as four words in series, like this:

```
KTAB is handled as K     + + + +     TAB KTAB
WLKTABHM is handled as WL     + + + +     KTABHM WLKTABHM
```

Now the four 'words' WL + + + + KTABHM and WLKTABHM can be sorted into alphabetical order before the dictionary is consulted, and returned to the original order after the dictionary has been completely read. If it turns out that WLKTABHM is a word in its own right, the three preceding 'words' will be discarded. If WLKTABHM is not a word in the dictionary, but KTABHM is, then WLKTABHM will be discarded, and the linguistic program will have to make the proper disposition of the copula-preposition group WL, the link + + + + , and the noun KTABHM.

In the hypothetical coding system, the linguist might specify the prefixes in this way:

```
$ PREFIX AL
$ PREFIX WAL
```

The two nouns KTAB (singular) and KTB (broken plural) have been called 'triptotes'. The opposite of 'triptote' in this context is 'diptote'; the only difference is that the form ending in A (accusative indefinite) does not occur in a diptote, and the form with 'zero' ending serves for all cases and states. Many broken plurals are diptotes, but diptote singulars never have regular plurals, so we need only one paradigm for all sorts of masculine diptotes:

```
$ PARADIGM 3 (FOR MASCULINE DIPTOTE SINGULARS AND BROKEN
  PLURALS)
-      NOM/ACC/GEN + DEF/CSTR/INDEF
Y      NOM/ACC/GEN+MY
H      NOM/ACC/GEN+HIS
HA     NOM/ACC/GEN+HER
NA     NOM/ACC/GEN+OUR
HM     NOM/ACC/GEN + THEIR
```

Feminine nouns are actually somewhat simpler in inflection than masculines. The singular form, e.g. BQRX (a cow), serves for all cases and states without a possessive suffix. The feminine ending X is changed to T before possessive suffixes. In the regular plural, the final X is changed to AT (e.g. BQRAT) for all cases and states, and this is also the plural form to which possessive suffixes are added. So we can establish another paradigm:

```
$ PARADIGM 4 (FOR FEMININE NOUNS WITH SOUND PLURALS)
X      NOM/ACC/GEN + DEF/CSTR/INDEF + SING
AT     NOM/ACC/GEN+DEF/CSTR/INDEF+PLUR
TY     NOM/ACC/GEN+SING + MY
TH     NOM/ACC/GEN+SING + HIS
THA    NOM/ACC/GEN+SING + HER
TNA    NOM/ACC/GEN+SING + OUR
THM    NOM/ACC/GEN + SING + THEIR
ATY    NOM/ACC/GEN+PLUR + MY
ATH    NOM/ACC/GEN+PLUR + HIS
ATHA   NOM/ACC/GEN+PLUR + HER
ATNA   NOM/ACC/GEN+PLUR + OUR
ATHM   NOM/ACC/GEN + PLUR + THEIR
```

But as with masculine nouns, the great majority of feminine nouns take broken plurals. And indeed most feminine singular nouns have broken plurals which are masculine in form,

while some masculine singulars have broken plurals which are feminine in form. This does not cause any trouble, as it happens. But we need another paradigm for singular nouns that are feminine in form and have no regular plural, and for broken plurals that are feminine in form:

```
$ PARADIGM 5 (FOR SINGULARS AND BROKEN PLURALS THAT ARE FEMI-
  NINE IN FORM)
X       NOM/ACC/GEN+DEF/CSTR/INDEF
TY      NOM/ACC/GEN+MY
TH      NOM/ACC/GEN+HIS
THA     NOM/ACC/GEN+HER
TNA     NOM/ACC/GEN+OUR
THM     NOM/ACC/GEN+THEIR
```

Adjectives

Adjectives are inflected just like nouns, in principle. However, they are not often used in the construct state, or with possessive suffixes attached to them, so I shall ignore these possibilities for simplicity. The following paradigm could be used for adjectives that have sound plurals:

```
$ PARADIGM 6 (ADJECTIVES WITH SOUND PLURALS)
-       MASC+SING+(NOM/ACC/GEN+DEF)/(NOM/GEN+INDEF)
A       MASC+SING+ACC+INDEF
WN      MASC+PLUR+NOM+DEF/INDEF
YN      MASC+PLUR+ACC/GEN+DEF/INDEF
X       FEM +SING+NOM/ACC/GEN+DEF/INDEF
AT      FEM+PLUR+NOM/ACC/GEN+DEF/INDEF
```

For adjectives without sound plurals, this paradigm could be used:

```
$   PARADIGM 7 (ADJECTIVES WITHOUT SOUND PLURALS)
-       MASC+SING+(NOM/ACC/GEN+DEF)/(NOM/GEN+INDEF)
A       MASC+SING+ACC+INDEF
X       FEM + SING+NOM/ACC/GEN+DEF/INDEF
```

and for broken plurals of adjectives, this paradigm:

```
$   PARADIGM 8 (BROKEN PLURALS OF ADJECTIVES)
-       MASC/FEM+PLUR+(NOM/ACC/GEN+DEF)/(NOM/GEN+INDEF)
A       MASC/FEM+PLUR+ACC+INDEF
```

Broken plural adjectives which have the feminine ending, or are diptote, are invariable words.

Verbs

As it is ordinarily printed, the conjugation of an Arabic verb is not very complicated, and in a sense there are no irregular verbs. But the roots which are traditionally called ‘weak’ give rise to verb-classes which are somewhat analogous to the conjugations of a Romance language. In addition, verbs can take direct object suffixes, just as nouns can take possessive suffixes. Finally, the forms of the imperfect tense are inflected at both ends.

Here is the conjugation of a 'strong' verb. Notice that the form KTB can mean either 'he wrote' or 'books'. This kind of homography is very common in Arabic. Homophonies are much less common, but the suppression of short vowels in printing produces many homographies.

Perfect tense

KTB	(he wrote)	KTBWA	(they wrote)
KTBT	(she wrote)		
KTBT	(I wrote)	KTBNA	(we wrote)

Imperfect tense

YKTB	(he writes)	YKTBWN	(they write)
TKTB	(she writes)		
AKTB	(I write)	NKTB	(we write)

The participles are KATB (active) and MKTWB (passive), but it is likely that participles would have to be carried in the dictionary as separate entries; so they are not included in this paradigm. The infinitive does not exist in Arabic (subordinate clauses and verbal nouns are used instead).

The imperfect as given above is actually the imperfect indicative; there are also an imperfect subjunctive and an imperfect jussive; but in the normal printed forms they differ from the indicative only in having YKTBWA instead of YKTBWN. It would therefore be convenient for the linguist to have codes INDIC, SUBJ, and JUSS available, but to have a rule that IMPF without any of them was equivalent to IMPF+INDIC/SUBJ/JUSS.

The paradigm must really be expanded, to take care of the direct object suffixes:

KTB	(he wrote)	KTBWA	YKTB	YKTBWN	YKTBWA
KTBH	(he wrote him)	KTBWH	YKTBH	YKTBWNH	YKTBWH
KTBHA	(he wrote her)	KTBWHA	YKTBHA	YKTBWNHA	YKTBWHA
KTBNY	(he wrote me)	KTBWNY	YKTBNY	YKTBWNNY	YKTBWNY
KTBHM	(he wrote them)	KTBWHM	YKTBHM	YKTBWNHM	YKTBWHM
KTBNA	(he wrote us)	KTBWNA	YKTBNA	YKTBWNNA	YKTBWNA

KTBT	TKTB
KTBTH	TKTBH
KTBTHA	TKTBHA
KTBTNY	TKTBNY
KTBTHM	TKTBHM
KTBTNA	TKTBNA

KTBT	KTBNA	AKTB	NKTB
KTBTH	KTBNAH	AKTBH	NKTBH
KTBHA	KTBNAHA	AKTBHA	NKTBHA
KTBTHM	KTBNAHM	AKTBHM	NKTBHM

But since machine translation paradigms of the type I am proposing cannot handle the prefixes Y T A and N, the verbal paradigm will have to be somewhat simpler:

```
$   PARADIGM 11 (STRONG VERBS)
-        PERF+HE
H        PERF+HE + HIM
HA       PERF + HE + HERR          (HERR is arbitrarily used as a symbol for the
NY       PERF+HE + ME                third person feminine direct object suffix, in
NA       PERF+(HE + US)/WE           contrast to HER for the third person
T        PERF+SHE/I                  feminine possessive)
TH       PERF+SHE/I + HIM
THA      PERF+SHE/I+HERR
TNY      PERF + SHE + ME
THM      PERF +SHE/I + THEM
TNA      PERF + SHE + US
WA       PERF+THEY
WH       PERF+THEY+HIM
WHA      PERF +THEY + HERR
WNY      PERF + THEY + ME
WHM      PERF+THEY+THEM
WNA      PERF + THEY+US
NAH      PERF + WE + HIM
NAHA     PERF + WE + HERR
NAHM     PERF+WE+THEM
WN       IMPF+INDIC + THEY
WNH      IMPF+INDIC+THEY + HIM
WNHA     IMPF+INDIC+THEY+HERR
WNNY     IMPF+INDIC+THEY+ME
WNHM     IMPF+INDIC+THEY+THEM
WNNA     IMPF+INDIC+THEY+US
```

Since the paradigm does nothing for the prefixes of the imperfect, it is necessary to define the prefixes as follows:

```
$    PREFIX    Y
$    PREFIX    T
$    PREFIX    A
$    PREFIX    N
```

And in fact an imperfect form can be further prefixed with the adverb S (indicating future time) or the conjunction L (indicating purpose); and either the conjunction F or the copula W can be still further prefixed. So prefixes SY ST SI (not SA) SN LY LT LI LN WY WT WI WN FY FT FI WSY WST WSI WSN must also be defined.

Then when the word YKTBHM occurs in the text, it will be read as Y + + + + KTBHM YKTBHM; YKTBHM will not be found in the dictionary, but KTBHM will be; and there must be a linguistic operation to delete Y + + + + from the sentence, but in exchange to alter PERF+HE+THEM to IMPF+HE+THEM in the grammatical coding obtained for KTBHM.

Many verbs are conjugated like KTB except for a single difference; the perfect forms all

have the A prefix. Such verbs are appropriately assigned to the same paradigm as KTB, because their endings are identical. But a special code—say PFPX—should be given to their stems in the dictionary, by which the interpretation of an A prefix can be modified.

Weak verbs

Several more 'conjugations' for Arabic verbs result from phonetic alterations. In describing them, I shall ignore the question of the direct object suffixes, since they are added to the verbal forms with perfect regularity.

'Doubled' verbs

Perfect Tense		*Imperfect Tense*					
		Indicative		*Subjunctive*		*Jussive*	
MD	MDWA	YMD	YMDWN	YMD	YMDWA	YMDD	YMDWA
MDT		TMD		TMD		TMDD	
MDDT	MDDNA	AMD	NMD	AMD	NMD	AMDD	NMDD

If we allow the character * in a paradigm to stand for any character (or digraph) identical with the preceding one, this can be handled by the paradigm:

```
$   PARADIGM 12 (DOUBLED VERBS)
-       PERF + HE
T       PERF+SHE
*T      PERF + I
WA   PERF+THEY
*NA  PERF+WE
WN   IMPF + INDIC + THEY
*       IMPF+JUSS +HE/SHE/I/WE
```

'Hollow' verbs

Perfect Tense		*Imperfect Tense*					
		Indicative		*Subjunctive*		*Jussive*	
QAL	QALWA	YQWL	YQWLWN	YQWL	YQWLWA	YQL	YQWLWA
QALT		TQWL		TQWL		TQL	
QLT	QLNA	AQWL	NQWL	AQWL	NQWL	AQL	NQL

This kind of verb will require three entries in the dictionary, for stems (e.g.) QAL, QWL, and QL; and three paradigms have to be provided:

```
$   PARADIGM 131 (HOLLOW VERBS—HOLLOW PERFECT FORMS)
-       PERF+HE
T       PERF + SHE
WA   PERF+THEY
$   PARADIGM 132 (HOLLOW VERBS—HOLLOW IMPERFECT FORMS)
-       IMPF + INDIC/SUBJ + HE/SHE/I/WE
WN   IMPF + INDIC+THEY
WA   IMPF+ SUBJ/JUSS +THEY
$       PARADIGM 133 (HOLLOW VERBS—NON-HOLLOW FORMS)
-       IMP + JUSS + HE/SHE/I/WE
T       PERF+I
```

'Defective' verbs with third radical ya

Perfect Tense		*Imperfect Tense*					
		Indicative		*Subjunctive*		*Jussive*	
RMA/	RMWA	YRMY	YRMWN	YRMY	YRMWA	YRM	YRMWA
RMT		TRMY		TRMY		TRM	
RMYT	RMYNA	ARMY	NRMY	ARMY	NRMY	ARM	NRM

These forms can be handled by a single dictionary entry with the stem RM, and a paradigm like this:

```
$  PARADIGM 14 (DEFECTIVE VERBS IN YA)
A/      PERF+HE
T       PERF+SHE
YT      PERF+I
WA      PERF+THEY
YNA     PERF+WE
Y       IMPF+INDIC/SUBJ+HE/SHE/I/WE
WN      IMPF+INDIC+THEY
        IMPF+JUSS + HE/SHE/I/WE
```

'Defective' verbs with third radical waw

Perfect Tense		*Imperfect Tense*					
		Indicative		*Subjunctive*		*Jussive*	
NDA	NDWA	YNDW	YNDWN	YNDW	YNDWA	YND	YNDWA
NDT		TNDW		TNDW		TND	
NDWT	NDWNA	ANDW	NNDW	ANDW	NNDW	AND	NND

These forms can be handled by a single dictionary entry with the stem ND, and a paradigm like this:

```
$  PARADIGM 15 (DEFECTIVE VERBS IN WAW)
A       PERF+HE
T       PERF+SHE
WT      PERF+I
WA      PERF+THEY
WNA     PERF+WE
W       IMPF+INDIC/SUBJ +HE/SHE/I/WE
WN      IMPF+INDIC+THEY
        IMPF+JUSS+HE/SHE/I/WE
```

There are a few more conjugation patterns that would have to be accounted for by a translation system; most of them involve stem changes that would be handled similarly to the stem changes of hollow verbs.

I noted earlier that, for example, KTBHA could mean either ‘he wrote her’ or ‘her books’. This kind of homography is only an occasional annoyance in most European languages, but it is so prominent a feature of Arabic that an extra feature in the dictionary

look-up program would be required. There would undoubtedly be two entries in the dictionary, which I may summarize thus:

KTB = P2 NOUN+MASC+PLUR *BOOK $S
KTB = P11 VERB *WRITE $S $/PAST/WROTE $/PPL/WRITTEN

The word KTBHA would be looked up according to both these entries, and indeed any others in the dictionary that would fit it. If there were only these two entries that fit it, then the program would handle it as though there were two words KTBHA in the sentence, with a linking 'word' / / / / between them. We might then represent what the dictionary would produce for KTBHA thus:

KTB = HA NOUN+MASC+PLUR+NOM/ACC/GEN+HER *BOOK $S
/ / / / LINK
KTB = HA VERB+PERF+HE+HERR *WRITE $S I/PAST/WROTE $/PPL/ WRITTEN

For the word YKTBHA the situation would be still more complicated; before the dictionary was looked at, this would have been expanded into four 'words': Y + + + + KTBHA YKTBHA and the look-up program would afterwards produce:

Y PREFIX
+ + + + LINK
KTB = HA NOUN+MASC+PLUR+NOM/ACC/GEN+HER *BOOK $S
/ / / / / LINK
KTB = HA VERB+PERF + HE+HERR *WRITE $S $/PAST/WROTE $/PPL/ WRITTEN
YKTBHA NOT IN DICTIONARY

After dictionary look-up, there has to be a linguistic operation that chooses to retain, e.g. either Y + + + + KTB = HA / / / / KTB = HA or YKTBHA. Somewhat later in the process another linguistic operation must choose to retain only one out of each group of alternatives linked by / / / / .

Arabic is so full of these difficulties that at first sight the situation might look as nearly hopeless as for machine translation of Babylonian cuneiform. A further discouragement is that although nouns are supposed to have three cases, it is evident from the description I have given that a case indication is very rarely visible in the printed text. The only obvious factor that retrieves the situation somewhat is that Arabic has a surprisingly rigid word order.

The business of case endings illustrates rather neatly the contrast between a conventional descriptive grammar and an algorithmic grammatical recognition procedure. If you add to a conventional description one statement: "The case endings of nouns are not, however, written except in the following uncommon situations. . . ." you do not spoil it as a description. But if you have to adjust a recognition procedure accordingly, you will probably have to discard it completely and invent a new one, much more complex than the old one.

3. MACHINE ANALYSIS OF ARABIC SYNTAX

In the last section I discussed 'morphology'. As far as the practice of machine translation is concerned, morphology is a branch of the dictionary look-up system. Considering each word of the input text separately, how can one extract all possible information from it?

Morphology cannot help us in resolving homographies; groups of words must be considered for this, and so we are brought to consider the syntax of Arabic before we have quite finished with morphology in the ordinary sense. To make a recognition procedure work, we are bound to mix up our levels of linguistic analysis; whereas in ordinary descriptive linguistics one makes every effort to avoid mixing levels.

I shall base this lecture on one Arabic sentence, chosen almost at random. What the sentence means is: "Locusts are divided into about two hundred species, but the species which cause the greatest damage have habits which are nearly similar in reproduction and development." In the transcription I have been using, the Arabic is:

YNQSM ALJRAD ALA/H/WALY MAITY NWC, ALA AN ALNWC ALD/Y YSBB ALD.RR ALJSYM, LH CADAT TKAD TTS/ABH FY ALTWALD WALNMW.

Now in the dictionary look-up process which I described, many of these words would be broken up by having a prefix taken from the left-hand end (e.g. YNQSM would be handled as four words, Y + + + + NQSM YNQSM) and some words would be looked up more than once, according to different entries in the dictionary (e.g. ALD.RR would first be broken up into AL + + + + D.RR ALD.RR; then D.RR would be looked up both as a noun and as a verb, with / / / / between them). I shall now represent schematically what might be obtained from the dictionary for the first ten words of the above Arabic sentence:

1.	Y	prefix
2.	+ + + +	link
3.	NQSM	VERB+PFPX+PERF+HE *BE DIVIDED
4.	YNQSM	not in dictionary
5.	AL	prefix
6.	+ + + +	link
7.	JRAD	NOUN + MASC + SING + (NOM/ACC/GEN + DEF/CSTR)/(NOM/GEN + INDEF) *LOCUST $S
8.	ALJRAD	not in dictionary
9.	AL	prefix
10.	+ + + +	link
11.	A/	not in dictionary
12.	ALA/	PREP *TO
13.	H/WALY	ADVB *APPROXIMATELY
14.	MAITY	NUMERAL + GEN/ACC + CSTR *TWO HUNDRED
15.	N	prefix
16.	+ + + +	link
17.	WC	not in dictionary
18.	NWC	NOUN + MASC + SING + (NOM/ACC/GEN + DEF/CSTR)/(NOM/GEN + INDEF) *KIND $S
19.	,	COMMA *,
20.	AL	prefix
21.	+ + + +	link
22.	A	prefix
23.	ALA	CONJ *BUT
24.	A	prefix
25.	+ + + +	link
26.	N	prefix
27.	AN	PARTICLE
28.	AL	prefix
29.	+ + + +	link
30.	NWC	NOUN + MASC + SING + (NOM/ACC/GEN + DEF/CSTR)/(NOM/GEN + INDEF) *KIND $S
31.	ALNWC	not in dictionary.

Now each + + + + in the list above is preceded by a prefix, and followed first by the de-prefixed residue of a word and then by the whole word. We need a linguistic operation that might be coded thus:

```
$ OPERATION ++CHOICE

          GO TO LEFT END
AAA       RIGHT TO    + + + +
          RIGHT
          QU NOT-IN-DICT      NXT-BBB
CCC       DELETE
          DELETE
DDD       DELETE              AAA
BBB       RIGHT
          QU NOT-IN-DICT      DDD-NXT
          LEFT                CCC
```

An operation consists of a series of commands; every command contains a function, and may have a 3-letter 'label' to the left of the function. To the right of the function one may specify which command to carry out next, or which to carry out next in case of a 'yes' answer to a question, and which in case of 'no'. NXT in this context means the next following command. Where nothing is specified to the right of the function, the next command is to be executed next; except that in case of a 'no' answer, the operation stops. Execution of an operation begins at the first command.

Operation '+ + CHOICE' will go through the whole sentence and examine the surroundings of each + + + + . Whenever the item next on the right, the residue of a de-prefixed word, is not in the dictionary, the prefix, link, and residue are deleted from the sentence, leaving only the original full word. If the residue is in the dictionary, the item next on its right is tested. This item is the original full word. If it was in the dictionary, it is retained and the residue, link, and prefix to its left are deleted. If the full word was not in the dictionary, it is deleted from the sentence, and the prefix, link, and residue are left. This gives us a sentence like this:

```
 1. Y              prefix
 2. + + + +        link
 3. NQSM           VERB + PFPX + PERF + HE *BE DIVIDED
 4. AL             prefix
 5. + + + +,       link
 6. JRAD           NOUN+ MASC +SING+(NOM/ACC/GEN + DEF/CSTR)/(NOM/GEN
                   + INDEF) *LOCUST $S
 7. ALA/           PREP *TO
 8. H/WALY         ADVB *APPROXIMATELY
 9. MAITY          NUMERAL+ACC/GEN + CSTR *TWO HUNDRED
10. NWC            NOUN+ MASC +SING+(NOM/ACC/GEN + DEF/CSTR)/(NOM/GEN
                   +INDEF) *KIND $S
11. ,              COMMA *,
12. ALA            CONJ *BUT
13. AN             PARTICLE
14. AL             prefix
15. + + + +        link
```

16. NWC	NOUN+ MASC +SING+(NOM/ACC/GEN+DEF/CSTR)/(NOM/GEN + INDEF) *KIND $S
17. ALD/Y	RELPRON + MASC + SING + NOM/ACC/GEN *WHICH
18. Y	prefix
19. +.+ + +	link
20. SBB	VERB + PERF + HE *CAUSE $S $D $XING
21. ////	link
22. SBB	NOUN + MASC + SING+(NOM/ACC/GEN+DEF/CSTR)/(NOM/GEN +INDEF) *REASON $S
23. AL	prefix
24. + + + +	link
25. D.RR	NOUN + MASC + SING+(NOM/ACC/GEN+DEF/CSTR)/(NOM/GEN + INDEF) *DAMAGE $S
26. ////	link
27. D.RR	VERB + PERF + HE *DAMAGE $S $D $XING
28. AL	prefix
29. + + + +	link
30. JSYM	ADJ + MASC + SING + (NOM/ACC/GEN + DEF)/(NOM/GEN + INDEF) *GREAT
31. ,	COMMA *,
32. L	prefix
33. + + + +	link
34. H	PRONOUN + MASC + SING + GEN
35. CADAT	NOUN + FEM + PLUR + NOM/ACC/GEN + DEF/INDEF/CSTR *HABIT $S
36. T	prefix
37. + + + +	link
38. KAD	VERB + PERF + HE *BE NEAR
39. ////	link
40. KAD	VERB + IMPF+INDIC/SUBJ + HE/SHE/I/WE *BE NEAR
41. T	prefix
42. + + + +	link
43. TS/ABH	VERB + PERF + HE *BE SIMILAR
44. FY	PREP *IN
45. AL	prefix
46. + + + +	link
47. TWALD	NOUN + MASC + SING + (NOM/ACC/GEN + DEF/CSTR)/(NOM/GEN + INDEF) *REPRODUCTION $S
48. WAL	prefix
49. + + + +	link
50.NMW	NOUN + MASC + SING + (NOM/ACC/GEN + DEF/CSTR)/(NOM/GEN + INDEF) *DEVELOPMENT $S

The items that resulted from the original word YSBB were more complicated than I have described so far: Y + + + + SBB //// SBB YSBB. Operation '+ + CHOICE' as laid out on page 17 would not have got rid of YSBB properly, but a somewhat more complicated operation, which took account of the possibility of encountering the link //// , could do it.

Before we can go very much further, we must make the choices implied by the / / / / links at positions 21, 26, and 39. Happily, all of these choices are easy to make because of the prefixes. Most prefixes will be found attached either to verbs exclusively or to nouns exclusively. The lists of prefixes I have given may not really be exhaustive even for modern printed Arabic, but the only non-specific prefixes they contain are the copula W and the conjunction F, which can be attached to any sort of word.

So let us suppose that every prefix requiring a noun (or adjective) after it is given, in its dictionary entry, the grammatical code NPFX, and every one that requires a verb is given VPFX. Then the following operation will clear up most of the choices:

```
$ OPERATION PREFIX-CHOICES
                GO TO LEFT END
        AAA     RIGHT TO   ////
                LEFT
                LEFT
                QU + + + +              NXT-BBB
                LEFT
                QU NPFX                 NXT-CCC
        AAA     RIGHT
                RIGHT
                QU NOUN/ADJ             DDD-NXT
        EEE     DELETE
                RIGHT
                DELETE                  AAA
        DDD     RIGHT
                RIGHT
                QU NOUN/ADJ             AAA-NXT
        FFF     DELETE
                DELETE                  AAA
        CCC     QU VPFX                 NXT-BBB
                RIGHT
                RIGHT
                QU VERB                 NXT-EEE
                RIGHT
                RIGHT
                QU VERB                 AAA-FFF
        BBB     RIGHT TO   ////         AAA
```

This operation makes two unwarranted assumptions: first, that there will never be more than two alternatives linked by / / / / , i.e. that no word will ever be looked up according to more than two entries in the dictionary; and second, that we will not get the unexpected case in which, e.g., a verbal prefix finds only alternative nouns after it. But applied to the above sentence, it would correctly delete items 21, 22, 26 and 27.

The operation would fail to make the choice required by the / / / / link at position 39. This choice could be made by another operation, which noted that the prefix T prefers an item with IMPF to an item with PERF, if an item with IMPF is available.

A series of operations has to be carried out to unite, or at least associate, the prefixes with the residues which follow them. Most notably, the verbal prefixes usually change the following residues from PERF to IMPF, and determine their persons. Noun prefixes involving the article AL will make the nominal or adjectival residues definite, and eliminate all indefinite or construct possibilities in their grammatical coding. The prefixes involving the prepositions B L and K, and the copula W and conjunction F, will leave those elements standing in the sentence as separate items. Let us take all this as done, and set out the sentence again:

```
1. Y + NQSM       VERB + IMPF + HE *BE DIVIDED
2. AL + JRAD      NOUN + MASC + SING + DEF + NOM/ACC/GEN *LOCUST $S
3. ALA/           PREP *TO
4. H/WALY         ADVB *APPROXIMATELY
5. MAITY          NUMERAL + GEN/ACC + CSTR *TWO HUNDRED
```

6.	NWC	NOUN + MASC + SING + (NOM/ACC/GEN + CSTR)/(NOM/GEN + INDEF) *KIND $S
7.	,	COMMA *,
8.	ALA	CONJ *BUT
9.	AN	PARTICLE
10.	AL + NWC	NOUN + MASC + SING + DEF + NOM/ACC/GEN *KIND $S
11.	ALD/Y	RELPRON + MASC + SING + NOM/ACC/GEN *WHICH
12.	Y + SBB	VERB + IMPF + HE *CAUSE $S $D *XING
13.	AL + D.RR	NOUN + MASC + SING + DEF + NOM/ACC/GEN *DAMAGE $S
14.	AL + JSYM	ADJ + MASC + SING + DEF + NOM/ACC/GEN *GREAT
15.	,	COMMA *,
16.	L	PREP *TO
17.	H	PRONOUN + MASC + SING + GEN
18.	CADAT	NOUN+ FEM + PLUR +NOM/ACC/GEN+ INDEF/CSTR *HABIT $S
19.	T + KAD	VERB + IMPF + INDIC/SUBJ + SHE *BE NEAR
20.	T + TS/ABH	VERB + IMPF + SHE *BE SIMILAR
21.	FY	PREP *IN
22.	AL + TWALD	NOUN + MASC + SING + NOM/ACC/GEN + DEF *REPRODUCTION *S
23.	W	COPULA *AND
24.	AL + NMW	NOUN + MASC + SING+NOM/ACC/GEN + DEF *DEVELOPMENT $S

Getting this far would involve a great deal of work, much of it surprisingly detailed, but nothing really abstruse. All we have done is to consult the dictionary, and to break up some of the words as printed into two or more word-like logical elements. If the input language were, say, Chinese, we would have had to carry out the opposite process of grouping typographical words into logical words.

Now the real problems begin. We have to find out, or at least guess, what the words are doing in the sentence; how they are related. As I have said several times, the nominative, accusative, and genitive cases are practically unmarked; but we shall do the best we can. I shall state some 'fairly good rules'; they work well on this sentence, and on many sentences, but they would obviously fail too often. However, it would be an entirely reasonable research procedure to put a set of such 'fairly good rules' into practice on a computer, apply them to a few thousand sentences, and examine the output to see what the next approximation should be.

Fairly Good Rule. 1. If a noun or numeral may be construct, and is followed by a noun that may be genitive, make the former exclusively construct in state, and the latter exclusively genitive in case, and consider the former to govern the latter.

The only effect of this in the above sentence is to join MAITY to NWC; NWC loses its possibilities of being nominative and accusative case. This rule could be coded thus:

```
$ OPERATION F.G.R.-1      GO TO LEFT END
                    BBB   RIGHT TO CSTR
                          RIGHT
                          QU NOUN              NXT-AAA
                          QU GEN               NXT-AAA
                          DELETE ACC
                          DELETE NOM
                          GOVERNED
                          LEFT
                          GOVERNOR
                          DELETE DEF
                          DELETE INDEF         BBB
                    AAA   LEFT                 BBB
```

Fairly Good Rule. 2. An adjective agrees with the nearest noun to its left if it has the same state (definite or indefinite; construct adjectives are rare). (Concord in gender and number between nouns and adjectives is rather weak; some things are 'forbidden', but no complete description of what is 'permitted' or 'required', under exactly what circumstances, seems to exist for the modern written language. Concord in case is hardly worth considering, since although strict in theory, it is so little apparent in practice.)

The only effect of this in the above sentence will be to join ALJSYM to ALD.RR. The rule could be coded thus:

```
$ OPERATION F.G.R.-2
                GO TO LEFT END
AAA             RIGHT TO ADJ
                CALL THIS ITEM K
                QU DEF                  NXT-BBB
                LEFT TO NOUN
                QU DEF                  NXT-CCC
DDD             MODIFIED
                GO TO ITEM K
                MODIFIER                *AAA
CCC             GO TO ITEM K             AAA
BBB             LEFT TO NOUN
                QU INDEF                DDD-CCC
```

Fairly Good Rule. 3. A preposition governs the nearest noun, numeral, or pronoun in the genitive case on its right, and makes it exclusively genitive in case. This means that in the sentence above, ALA/ governs MAITY, L governs H, FY governs ALTWALD.

The rule could be coded thus:

```
$ OPERATION F.G.R.-3
                GO TO LEFT END
AAA             RIGHT TO PREP
BBB             CALL THIS ITEM K
DDD             RIGHT
                QU PREP                 BBB-NXT
                QU NOUN                 CCC-NXT
                QU NUMERAL              CCC-NXT
                QU PRONOUN              CCC-DDD
CCC             QU GEN                  NXT-AAA
                DELETE NOM
                DELETE ACC
                GOVERNED
                GO TO ITEM K
                GOVERNOR                AAA
```

Fairly Good Rule. 4. An adjectival clause begins with each relative pronoun, and ends just before the next conjunction or punctuation; it modifies the noun next preceding the relative pronoun.

This rule would join ALD/Y YSBB ALD.RR ALJSYM to ALNWC as a modifier. It could be coded thus:

```
$ OPERATION F.G.R.-4
                GO TO LEFT END
AAA             RIGHT TO RELPRON
                CALL THIS ITEM K
CCC             RIGHT
                QU PUNCT            BBB-NXT
                QU RELPRON          BBB-NXT
                QU CONJ             BBB-CCC
BBB             LEFT
                END CLAUSE
                GO TO ITEM K
                BEGIN CLAUSE
                MODIFIER
                LEFT TO NOUN
                MODIFIED
                GO TO ITEM K        AAA
```

Actually, a relative pronoun will only occur after a definite noun; an adjective clause modifying an indefinite noun follows it immediately, with no explicit indication that a new clause is beginning. An instance of this in the sentence above is the clause beginning with TKAD, which modifies CADAT. (It is entirely typical of Arabic that while CADAT is plural, TKAD logically refers to it but is singular. However, the difficulties would be much the same even with a strict concord like that in most Indo-European languages.) The clue to such an adjectival clause is usually that a verb follows directly after an indefinite noun. This is neither a necessary nor a sufficient condition, but it is probably the best simple statement of the conditions. If one can determine that there is a predicate before the indefinite noun, the likelihood that the verb begins an adjectival clause is much greater. (The predicate before CADAT is the prepositional phrase L H, which has to be translated 'it has'.)

The sentence has to be divided into main clauses; conjunctions and punctuations would serve as indications in the sample sentence, but certainly this is an inadequate rule for general use. Within a clause, nearly every noun which is not governed by a preposition will be either a subject or a direct object (or a predicate in a verb-less clause). The normal word-orders are subject-verb-object and verb-subject-object, if all three elements are present. For a clause having either a subject noun or an object noun, but not both, the normal orders are subject-verb, verb-subject, and verb-object. The cases of nouns being virtually unmarked, the patterns 'verb-subject' and 'verb-object' appear to be hopelessly confounded, though the distinction between subject and object must be important for translation into any European language. This is a problem which no amount of theoretical or mathematico-linguistic research can do much to solve. The only thing to do is to examine as much Arabic text as possible, and try to discern the signals which must exist in the text. Perhaps there are no signals beyond the semantic facts—some nouns can be subjects to certain verbs, others cannot, and so on.

In any case, the easiest way to present a large quantity of such material to a human researcher would be to get a machine to translate a large body of text, using an inadequate rule. In the reading of the translation, every instance of a subject wrongly called an object, or an object wrongly called a subject, would immediately be evident to a human being, who

is blessed with intuition. The translation program might be modified so that no translation at all was produced for a sentence in which the subject/object difficulty did not appear to arise. In this way, the 'translation' program would be used as a device to help select the material for the linguist to consider.

When I began to write this lecture, I knew that Arabic syntax was comparatively simple, and that a good description was available. I had forgotten that a simple descriptive grammar does not necessarily indicate a simple recognition procedure. An ordinary descriptive grammar, whether it has been programmed or not, gives a system of patterns such that nearly any sentence in the language can be fitted into one of the patterns by someone who knows the language. A good Latin grammar provides a model for any good sentence in Latin; a teacher can take a Latin sentence and convincingly show a student how the sentence conforms to the grammar in the book. Or the grammar could be programmed and used to generate random Latin sentences—this has been done for English by Dr. Yngve with extremely interesting results. But in the absence of the teacher, a beginning student of Latin would be unable to fit the sentence to a grammatical pattern; in other words to parse it. The grammar doesn't work this way for anyone who doesn't know the language. A good beginning student of Latin could use the grammar to construct grammatical Latin sentences, but he couldn't read a page of Cicero even with grammar and dictionary at hand.

A machine working by any algorithms we can imagine will never understand what anything *means;* therefore it needs a very complicated sort of recognition grammar to accomplish what a human being can do with conventional grammar, plus intuition and experience. A recognition grammar will turn out to be a thousand times more complicated than a conventional descriptive grammar. Yet this is really an illusion. Common sense tells us that a descriptive grammar and a recognition grammar should somehow be equivalent, and that either should be transformable to the other. So they would be, if they were both complete. However, a fractional descriptive grammar gives an illusion of completeness and elegance, while a fractional recognition grammar just gives bad machine output. Descriptive linguists nowadays try to go beyond the ordinary morphological and syntactic levels, but with little enough to show for it as yet. In the meantime, descriptive grammars, at their best, are rigorous over a field not much wider than the field which the Arab grammarians of the early middle ages were able to describe with rigor.

Suppose that a conventional descriptive grammar of a written language contains one tenth of the whole truth. This tenth will be the most useful one for a human being, and the best one to begin by learning. Now the whole truth of descriptive grammar and the whole truth of recognition grammar are probably equivalent; but the tenth part of the recognition grammar which is directly equivalent to the accessible tenth part of the descriptive grammar will not take us very far. It may be that a tenth part of the total recognition grammar would be enough for satisfactory machine translation; but it will not be the same tenth which is produced by transforming the known tenth of descriptive grammar.

Our ignorance about every language is enormous, as soon as we are obliged to leave behind our intuitive knowledge. Machine translation research makes this painfully clear, while traditional linguistics allows us to forget it. The new knowledge which machine translation demands cannot be spun out of our minds; we have to study the texts with a new set of questions in mind. If I believe that a machine translation research effort ought to aim at producing an algorithm within two or three years, I do not suppose that two or three years will show tremendous progress. But to work on machine translation without producing and testing an algorithm is like studying the synthesis of proteins without trying to syn-

thesize some proteins. You may think that the knowledge gained by analysing proteins will be enough to keep you on the right track, but the chances are that you will be drifting farther and farther away from it.

4. MACHINE SYNTHESIS OF ENGLISH OUTPUT

For the sake of neatness, I said that my third lecture would be about machine analysis of Arabic, and my fourth would be about machine synthesis of English. This is an absurdly disproportionate division of time. By quibbling, one can say that machine translation consists entirely of the analysis of the input language, or that it consists entirely of the synthesis of the output language. But as we normally use the term, the synthesis of the output language is what happens after we have done all the analysis we can—morphological, syntactic, and perhaps semantic—on the input language. Assuming the output language to be English, the synthesis involves the re-arrangement of the items of the sentence into an English order, the inflection of English stems that were originally brought from the dictionary, the insertion and suppression of articles, the choice of English equivalents for polysemic words, especially prepositions, and the choice of English expressions for what the input language expresses by inflectional elements.

Let me suppose that the analysis of my sample Arabic sentence is complete as to syntax; in other words that I have succeeded in parsing it completely. It will now look something like this:

```
 1. YNQSM      VERB + IMPF + INDIC + HE                  VS  VP              *BE DIVIDED
 2. ALJRAD     NOUN + MASC + SING + NOM + DEF            VS  VP              *LOCUST $S
 3. ALA/       PREP                                          VP PN          *TO
 4. H/WALY     ADVB                                             PN          *APPROXIMATELY
 5. MAITY      NUMERAL + GEN + CSTR                             PN NN       *TWO HUNDRED
 6. NWC        NOUN + MASC + SING + GEN + INDEF                    NN       *KIND $S
 7. ,          COMMA                              ..................*,
 8. ALA        CONJ                                                *BUT
 9. AN         PARTICLE                           TS
10. ALNWC      NOUN + MASC + SING + ACC + DEF     TS  AN  RR          *KIND $S
11. ALD/Y      RELPRON + MASC + SING + ACC            AN RR RR        *WHICH
                                                  ((((((((((((((((((((((
12. YSBB       VERB + IMPF + INDIC + HE           VO      RR RR       *CAUSE $S $D $XING
13. ALD.RR     NOUN + MASC + SING + ACC + DEF     VO      RR      AN  *DAMAGE $S
14. ALJSYM     ADJ + MASC + SING + ACC + DEF              RR      AN  *GREAT
15. ,          COMMA                              ))))))))))))))))))))))  *,
16. L          PREP                               PN      RR          *TO
17. H          PRONOUN + MASC + SING + GEN        PN      RR
18. CADAT      NOUN + FEM + PLUR + NOM + INDEF        RR              *HABIT $S

19. TKAD       VERB + IMPF + INDIC + SHE              RR VV           *BE NEAR
20. TTS/ABH    VERB + IMPF + INDIC + SHE          VP      VV          *BE SIMILAR
21. FY         PREP                               VP          PN      *IN
22. ALTWALD    NOUN + MASC + SING + GEN + DEF                 PN      *REPRODUCTION $S
23. W          COPULA                                         PN      *AND
24. ALNMW      NOUN + MASC + SING + GEN + DEF                 PN      *DEVELOPMENT $S
                                                  ))))))) ))))))))))))))))
```

The parentheses enclose subordinate clauses, while the line of dashes indicates a complete syntactic cut. The columns of letters in the middle of the page are supposed to represent binary relations as follows:

VS verb to following subject
VP verb to preposition semantically important to it (e.g. 'are divided *into*')
PN preposition to noun it governs
NN noun or numeral to noun it governs
TS subject-introducing particle to subject
AN noun to adjective or relative pronoun modifying it
RR antecedent noun to a later pronoun which refers to it (which pronoun may be the implicit subject contained by a verb)
VO verb to direct object
W verb to a following verb in the imperfect indicative or subjunctive which it 'governs'

Here is a simple operation for shifting subjects that follow their verbs into positions ahead of the verbs:

```
$ OPERATION SUBJECT-INVERSION

          GO TO LEFT END
AAA       RIGHT TO VERB
          QU VS                          NXT-AAA
          SLIDE ON VS
          LEFT END FOR SHIFT
          QU AN                          NXT-BBB
          SLIDE ON AN
          QU RELPRON                     NXT-BBB
          RIGHT TO ))))))
BBB       RIGHT END FOR SHIFT
          LEFT TO VERB
          LEFT
          SHIFT TO RIGHT OF THIS ITEM
          RIGHT TO VERB                  AAA
```

Here is a simple operation for pluralizing English nouns whose Arabic originals either are plural or follow numerals; adding 'THE' before nouns if their Arabic originals were definite; and adding 'OF' before them if they are governed by other nouns:

```
$ OPERATION NOUN-SYNTHESIS

          GO TO LEFT END
AAA       RIGHT TO NOUN                  NXT-BBB
          QU DEF                         NXT-AAA
          LEFT ADD ENG ( (THE))          AAA
BBB       GO TO LEFT END
CCC       RIGHT TO NOUN
          QU PLUR                        NXT-DDD
          ADD -S
DDD       QU BOTTOM OF NN                NXT-CCC
          SLIDE ON NN
          QU NUMERAL                     NXT-EEE
          SLIDE ON NN
          ADD -S                         CCC
EEE       SLIDE ON NN
          LEFT ADD ENG ((OF) )           CCC
```

But as the adjectives in Arabic normally follow their nouns, we must first have moved them ahead of their nouns. Let us assume that whenever a noun is followed by more than one adjective, the adjectives are to keep the same mutual order after they have been moved ahead of the noun. Then this operation will do it:

```
$ OPERATION ADJECTIVE-REVERSAL
                    GO TO LEFT END
          AAA       RIGHT TO NOUN
                    QU AN                       NXT-AAA
                    SLIDE ON AN
                    QU ADJ                      NXT-AAA
                    RIGHT
                    FLAG
          BBB       LEFT
                    QU NOUN                     AAA-NXT
                    DELETE
                    LEFT TO NOUN
                    LEFT ADD DELETED ENG
                    RIGHT TO FLAG               BBB
```

I shall not attempt to show a routine for inflecting English verbs. Since there are only two tenses in Arabic (as in Russian—but a more accurate comparison would be to say that each of the Arabic tenses corresponds to one member of a perfective-imperfective pair of Russian verbs) it is obvious that what we translate into English tenses will involve auxiliary verbs and particles in Arabic. Negation of an English verb is mixed up with inflection, so negation and tense inflection would have to be handled together. Furthermore, the grammatical number of an English verb will often have to be made plural even when it translates an Arabic singular verb.

However, let me look in particular at the verb phrase TKAD TTS/ABH. This will have to be translated 'which are nearly similar', but I would like to consider only the transition from *BE NEAR *BE SIMILAR to *BE NEARLY SIMILAR. This is an operation peculiar to the verb KAD, so I want to code it in this verb's dictionary entry or entries, and keep it out of the list of operations of general usefulness. The verb has three stems, of which two happen to coincide: KAD for hollow perfect tense forms, KAD for hollow imperfect tense forms, and KD for non-hollow forms. Here is how the dictionary entry for the first of these stems might be coded:

```
KAD = P131   VERB   *BE NEAR
- LOCAL OPERATION, PRIORITY 800
-                   QU VV
-                   SLIDE ON VV
-                   QU FIRST ENGLISH ((BE) )      AAA-NXT
-                   DELETE ENGLISH
-                   SLIDE ON VV
-                   REPLACE ENGLISH BY DELETED ENGLISH
-                   LEFT ADD ENG ( (NEARLY))   STOP
-         AAA       DELETE FIRST ENGLISH
-                   DELETE ENGLISH
-                   SLIDE ON VV
-                   REPLACE ENGLISH BY DELETED ENGLISH
-                   LEFT ADD ENG ( (BE NEARLY)) STOP
```

Here I have suggested how a dictionary entry can contain a piece of program. Note that to this 'local operation' a priority of 800 has been assigned. So far I have said nothing about how the linguist is to ensure that the operations involved in the translation process are performed in the order he wants. A flexible method for sequencing operations is to assign to each one a priority number within a fixed range, say 1 to 999. Then all the operations called for by a sentence will be executed in order of increasing priority number.

It is obvious how and when a 'local operation' like the one above will be brought into play. What about the operations that have to be carried out on every sentence? The following device has several advantages of convenience: Let every sentence in the input text be considered to begin with an extra word at its left-hand end, 'DUMMY'. Though this word is not really in the sentence, an item corresponding to it can be used to contain information about the sentence as a whole, and as a sentinel to mark the left-hand end of the sentence (it can also mark the right-hand end, if the sentence is laid out as a simple ring-shaped list). We are then allowed to have an entry in the dictionary for DUMMY, and this entry may contain instructions calling on all those operations which are to be used on every sentence:

```
DUMMY   XXX   XXX
- INSTRUCTION OPERATION ((++CHOICE)) PRIORITY 50
- INSTRUCTION OPERATION ((PREFIX-CHOICES)) PRIORITY 60
- INSTRUCTION OPERATION ((F.G.R.-l)) PRIORITY 100
```

and so on.

If a dictionary entry can call on an operation in this way, we can arrange for an operation to be called on not for every sentence, but only for sentences containing certain words. For instance, there may be a class of verbs in Arabic which are almost always followed by the preposition CLA/. It is worth recognizing the connection between such verbs and CLA/, in order to help account for the preposition. So we may code an operation to do this, call it OPERATION VERB-CLA/, and then put an instruction like the following into the dictionary entry for every such verb:

```
- INSTRUCTION OPERATION ((VERB-CLA/)) PRIORITY 300
```

Further, consider that after some verbs, CLA/ will be translated 'on', after others 'to', after others 'from', and so on. One might allow an instruction to include not only the name of an operation and a priority number, but also, optionally, one or more English words which the named operation could use if necessary. Then the dictionary entry for one of these verbs might contain

```
- INSTRUCTION OPERATION ((VERB-CLA/)) PRIORITY 300   *TO
```

the entry for another might contain

```
- INSTRUCTION OPERATION ((VERB-CLA/)) PRIORITY 300   *FROM
```

and so on.

The imaginary coding language which I have used in all these examples may not seem very attractive to linguists as a vehicle for expressing their ideas on machine translation. Partly this is due to incidental features, but partly it is due to the initial obscurity of anything that looks like computer programming. However, any linguist who wants to work seriously in the field has to face the problem of communicating with the programmers. One way of communicating is by verbal statements, but these will soon begin to resemble

statutes on taxation and inheritance—and instead of giving employment to generations of lawyers, these linguistic statements will only give headaches to programmers and linguists. Or the linguist can make flow-charts. But flow-charts are not as straightforward as they look, and the linguists will have to train themselves to be rigorous, and work for months with the programmers before all the necessary understandings are reached. Furthermore, the programmers cannot write a program that will implement flow-charts directly. In the long run, it is best for the linguists, or at least some of them, to invest their time in learning to express themselves in a language of rigid procedural statements, and for the programmers to invest their time in the complex programming needed to implement these statements.