COMPUTER-AIDED TRANSLATION OF PRE-EDITED NATURAL LANGUAGE SENTENCE STRUCTURES

T. G. CHANDRA VADIVELU*

(*E.C.E. Department, Indian Institute of Science, Bangalore*)

Received on October 15, 1976; Revised on May 12, 1977

ABSTRACT

English and other European languages have a common structure in sentence formation. In Indian languages unlike European languages, in most sentences the verb comes as the last element. Therefore, in translating English into any Indian languages the interlinguistic transformation is to be achieved.

In this paper Tamil is taken as the target language and structural transformation is done, using computer, before translation. The addition of endings after translation is also suggested. The use of pre- and post-editing would help to make unambiguous translation.

Key words : Machine translation, Indian languages, Computer analysis, Applied linguistics.

INTRODUCTION

This report is a continuation of a paper by P. C. Ganeshsundaram,[1] Part I: Theoretical Basis. The section numbers in this report are, therefore, a continuation of those in that paper.

In what follows we are concerned with the linear ordering of syntactic elements and their transposition from English into Tamil, when we have only one C-structure[2] in English.

2.4. *Transposition for Tamil*

Under the major plan of our work, the present (limited) project has been conceived of as a minor detour, taking Tamil here as the target language. Accordingly in this undertaking Tamil is written in Roman spelling

* Present address : Electronics and Communication Engg. Dept., College of Engineering, Madras 600 025.

and syntactic demarcations are taken up for computer handling. It is proposed to go step by step in handling highly complex structures. We discuss in this paper only sentences in English with one C-structure. That is each sentence analysed contains only one verb.

In trying to set up syntactic demarcation for Tamil from English, the following general principles are observed:

1. A pre-editor makes the demarcation of syntactic structures in English, and punches cards.[1,2,3]
2. Among the P-structure the subject, direct object, and indirect object, are the only ones treated as the main components and placed within the brackets ( ).
3. All other compliments of the verb are treated as additional parenthetical indications, and as such are placed within the brackets ⟨ ⟩, which are themselves placed between hyphens -( )-.
4. The computer is programmed to rearrange the successive main elements in the following way. [Even though there are many ways in which the P-structures could be transposed, the one suggested below was found to work well for most of the simple sentences.]

Since we are going to consider only one C-structure in this report, we can assume that all P-structures are placed between round brackets ( ). The P-structures within the sentence are divided into two groups, one before the verb and the other after the verb. The P-structures before the verb are retained in the transposed structure as such and the P-structures after the verb are just reversed in their order. In this process the last P-structure comes first immediately after the initial P-structure, and the verb goes to the last position. Let us consider one example.

⟨I⟩ SAW ⟨A SNAKE⟩ ⟨YESTERDAY⟩

In this sentence we have three P-structures and one verb. It can be split in the following way:

P_1 — I

V — saw

P_2 — a snake

P_3 — yesterday

In the sentences selected, the most convenient transposition before translation, so that the transposed words satisfy the synthetic structure requirement for Tamil, is given below:

$$S_E - P_1 V P_2 P_3$$

The transposed Tamil syntactic equivalent structure is

$$S_{ST} - P_1 \quad P_3 \quad P_2 \quad V$$

Before translation the English C-structure sentence is transposed as shown below:

I YESTERDAY A SNAKE SAW

which after translation gives[4]

NAANH NEERRHU ORU PAAMPU PAARTTEENH

2.5. *Programming for a Single-C-structure*

The computer is first allowed to read the sentence and the position of various syntactic markings are noted down. The C-structure markings are round brackets and the P-structure markings are angular brackets. A location L (12, 2) is allotted for this storage. The programme to accommodate ending will be reported later.

The programme has provision for four C-structures and so the C-structure marking positions are stored in the initial 4×2 locations. L (1, 1) to L (4, 1) stores 4 open structure position and L (1, 2) to L (4, 2) stores 4 closed structure positions. Next 6×2 locations are allotted for P-structure markings. L (5, 1) to L (10, 1) for P-structure open markings and L (5, 2) to L (10, 2) for close structure markings. The remaining 2×2 positions are allotted for special symbols like +, &, =, and . (dot).

(i) *Main Programme*

The various methods of allotting the nouns and other grammatical elements will be given later. For our present limited work the dictionary is allotted $12 \times 18 \times 11$ positions. Each word consists of twelve alphabetic symbols. Nouns are classified into three groups of eighteen words each. The other grammatical elements—article, verb, adverb, adjective, pronoun, conjunction, interjection and preposition—are given eighteen words each. Thus we get 12 alphabetic symbols $\times$ 18 words $\times$ 11 sets. In order to describe the main logic of translation let us consider the flow chart as given in Fig. 1.

The programme is made to read the dictionary. It then reads the text, separates a sentence from the text by noting down the position of

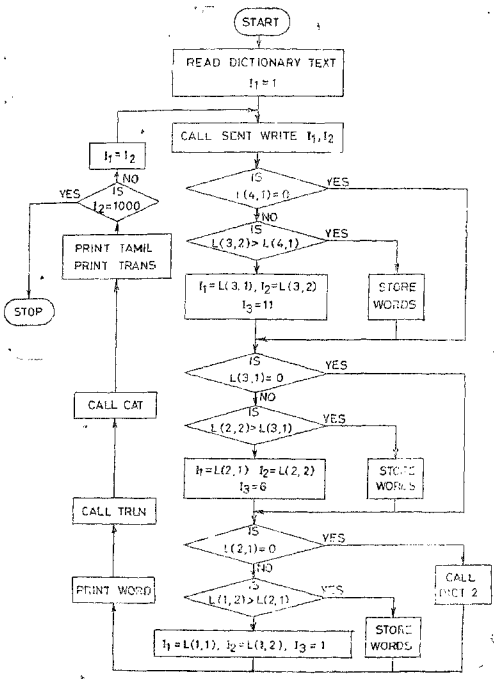

FIG. 1. Main programme; Programme for store words to be developed.

which a dot occurs. I1 = 1 is the initial value and a sub-routine SENT is called to find the next position of a dot and its position is noted, whose value is I2. I2 is returned to the main programme. Thus a sentence is available between I1 = 1 and I2 = 32 as shown below. These values are asked to be printed.

I = 1 ⟨*I* b⟩ SAW b ⟨A b SNAKE b⟩ ⟨YESTERDAY⟩. I2 = 32

Next we call for a sub-routine structure analysis. STRUC will read the sentence between I1 and I2 and will note down the position of syntactic elements. The various positions at which it occurs are stored in L (12, 2) as stated earlier and returned to the main programme.

Then the main programme checks whether L (4, 1) equals O, if not the words are stored in words dimension WORD (12, I3). Since this C-structure is the innermost C-structure we allot the location WORD (12, 16) to WORD (12, 20), *i..e*, the central portion. If L (4, 1) = 0, then the programme goes to check L (3, 1), if it is zero it goes on again to check L (2, 1) and so on. At level three if L (3, 1) is not zero we make a check on L (3, 1) and L (3, 2) to decide whether this includes L (4, 1) to L (4, 2). That is to say we have two alternatives, *viz*., the fourth structure may be inside the 3rd C-structure or the third C-structure lies before the fourth.

(	(	)	)	STRUCTURE I
L (3, 1)	L (4, 1)	L (4, 2)	L (3, 2)	
(	)	(	)	STRUCTURE II
L (3, 1)	L (3, 2)	L (4, 1)	L (4, 2)	

If it is structure one (I), we allot the elements lying between L (3, 1) and L (4, 1) to 10 to 15 locations of words and the other portion L (4, 2) and L (3, 2) to positions 21 to 25. Next we make a check whether L (3, 1) is zero; if not we give it to a format which stores the initial portion in 5 locations starting from 6 and the latter portion between 21 and 25. This is continued until we get a condition where L (2, 1) = 0; it means L (3, 1), L (4, 1) are all zero and we have L (1, 1) to L (1, 2) as the position at which only one C-structure is available (by virtue of the method of STRUC). In such a case we call DICT 2 which is the major work of this limited project. This programme has provision thus for 4 C-structures. Until we develop a complete syntactic transposition method for more than one C-structure we cannot go with the further simplification of these steps.

The main programme, then, prints the word. We have allotted tentatively five locations on the word and so we will have vacant locations at this stage. We then call for translation sub-routine TRLN which gives TRANS (12, 30) which contains the Tamil lexical equivalent of the WORD (12, 30). We will be having vacant locations which will also be printed with Trans. To eliminate this, we call for concatenation using CAT (TRANS, TAMIL). Tamil is given a continuous index over the sentence with only one blank between words. T (300) is the dimension allotted to this purpose and the value of I3, the end of Tamil (300) is returned to the main programme. When the value of I2 = 1000, we will terminate the programme and stop. Otherwise I1 value is made I2 to start for the next sentence and the programme is started from the beginning. Before going into any further discussion on the ending supply, we will see the various sub-routines,

(ii) *Sub-routine SENT*

In a given text any sentence is situated between two dots or a dot and a question mark. With I1 = 1 sub-routine SENT reads text, notes down the position at which it encounters a dot and finds this value L2 and it is returned to the main programme.

(iii) *Sub-routine STRUC*

This sub-routine analyses the syntactic markings of the given sentence between I1 and I2. It notes down the position of the various syntactic markings '('-C-open structure, ')'-C-close structure, '<'-P-open structure and '>'-P-close structure. When this sub-routine is called it notes down the positions at which these markings occur in L (12, 2) array. The C-structure markings are stored in L (K, 1) and L (K1, 1) for open and closed structure. The next L (K2, 1) and L (K3, 2) stores the P-structures where K2 and K3 start with 5. Because K2 and K3 start with 5, the beginning 4 locations are available for C-structures. K4 = 1 and K5 = 1 are allotted for + (plus) and = (equal) signs in the second variables of L (11, K4) and L (12, K5). K4 = 2 and K5 = 2 are two more locations for the special signs.

(iv) *Sub-routine DICT* 2

This sub-routine is used for forming the word for sentences with one C-structure. The main programme checks the values from L (I, J) and if L (2, 1) = 0 (*i.e.*, there is only one round bracket) and so only one C-structure it calls for this sub-routine. The syntactically transposed words are stored in WORD (12, 30). This means that we can store 30 words with each word having 12 alphabetic symbols. Every word is to be punched with a blank at the end. This will facilitate the programme to change its location number for every blank.

When the programme encounters a JCL, the next element should be JOP. If it is not JOP then a verb should be available between JCL and JOP. This verb should be sent to the last location. For the storage of these verb elements the last three word locations 28–30 are allotted.

Since we must keep count of the JOP after the verb and before the verb, we use index J for this purpose. The programme allots initial five locations for J = 0. If after the verb there is a P-structure J is incremented by 1. When J = 1 it is stored in 23 to 27 locations. Thus the words in locations 23 to 27 are the first P-structure after the verb. If a P structure is encountered for a second time the running index I3, on WORD

(I2, I3), is given value 18 to 22. If a third structure is encountered it is stored between I3 = 14 and 17 and so on, getting a complete inversion between the beginning of the verb to the end of the sentence.

(v) *Sub-routine TRLN*

This will translate the transposed sentence available in locations WORD (12, 30) to TRANS (12, 30) comparing the words transposed, with the dictionary. Location B (12) is loaded with the first word and C (12) takes in a word from the dictionary. Then letter by letter B (12) and C (12) are compared. If they are identical then the lexical determinant is outputed If it does not coincide with words available in the dictionary the same word is outputed to Trans for storage. This is essential because we need not store proper names or certain abbreviated names in the dictionary. For comparing B (12) with C (12), function I SAME is used. When the value returned by function I SAME is 1 we can say B (12) and C (12) are identical. This means the word in the transposed sentence is equal to DICTI (12, K3, K2). Then the word in location DICTI (12, K3, K2 + 11) is the Tamil equivalent. It is stored in TRANS (12, K1). The number added is 11 because we have 11 sets of 18 words for English arranged in serial order before 11 sets of 18 lexical Tamil equivalents for the first set.

(vi) *Function I SAME*

If the value returned by this function is 1, we can say that the word from the sentence matches with a word DIGTI (12, K3, K2). B (12) and C (12) are compared character by character and, if they are the same, we make I SAME = 1. Before the start of the programme I SAME = 0.

(vii) *Sub-routine CAT*

It is used to eliminate the blank in between words. We have allotted 30 words to fill up the blanks by transposition of the given sentence. If the sentence selected for translation has fewer than 5 words per structure and fewer than four structures, we will be left with blanks. Thus TAMIL (I3) will be storing characters in a word, starting from the first. If it encounters a BLA we introduce a blank for Tamil and increment the word number. If it encounters a blank at the beginning of the word itself, we do not introduce a blank. Therefore Tamil stores continuously the various words with blanks (only one) in between words.

2.6. *Dictionary*

As this is a small project we include in the dictionary all words selected out of 20 sentences. Even though we have just used all words and their

lexical equivalents we have not included the method for the endings. It is proposed to give a procedure for ending supply.

2.7. *Prospects*

As this is a small pilot project our dictionary is not complete, and our aim is simply to propose the method to be taken up later on a larger scale with a more complete dictionary.

We can classify the ending supply into two classes:

(i) prepositional phrase ending,

(ii) grammatical ending.

(i) *Prepositional Phrase Ending*

In this all simple prepositional elements will carry + mark when we devise a sub-routine that will read Tamil (300) and whenever it encounters a + sign that word is delayed and reintroduced after the noun which follows the preposition.

(ii) *Grammatical Ending*

(*a*) This will analyse the subject, object and the verb.

(*b*) Nouns, pronouns, verbs should carry a tag number for one character in length. It is appended to the Tamil equivalents at the translation phase.

The special numbers indicate the position at which ending is to be supplied.

A simple table will decide the correct ending for each special number which is included with Tamil.

Therefore the pre-editor must also give one (*) star to the direct and indirect object leaving a space in between the noun or pronoun and star *. These stars are replaced by their grammatical special numbers at the translation phase. English verbs are available at word (12, 28 or 29, 30) which can be analysed separately to find a grammatical number.

These three numbers should specify the correct Tamil ending which is appended to the subject and object.

CONCLUSION

It takes nearly 10 minutes to process the 20 given sentences.

The pre-editor must be trained to know in addition to structure marking, the use of star as mentioned earlier. This is really essential, not only to eliminate ambiguity but also to limit the infinite variety of structures to just a few simple finite varieties of sentences.

ACKNOWLEDGEMENTS

The author is grateful to Dr. P. C. Ganeshsundaram, FLS, Indian Institute of Science, for the overall guidance and encouragement in this project. He also takes this opportunity to express his sincere thanks to Mr. G. C. Kothari, E. E. Department, I.I.Sc., for fruitful discussions and to Mr. S. Namperumal, Computer Centre, I.I.Sc., for his help and guidance.

REFERENCES

1. Ganeshsundaram, P. C. — A practical approach to machine-aided translation with pre- and post-editing, Parts I and II, *Journal of the Indian Scientific Translators Association*, 1973, **2**, 47–60.

2. Ganeshsundaram, P. C. — Structural relativity in languages—Paper presented at the IV International Congress of Applied Linguistics, Stuttgart, August, 1975. Also in *Journal of the Indian Institute of Science*, 1975, **57**, 329–357.

3. Ganeshsundaram, P. C. and (Miss) Maya Devi — Syntactic description of bilingual patterns, *Journal of All India Institute of Speech and Hearing*, 1974–75, **5** and 6, 79–90.

4. Ganeshsundaram, P. C. — The development of a suitable script—An input problem in the analysis of Indian languages using IBM 360/44 computer. *Journal of the Indian Institute of Science*, May 1976, **58** 225–237.