[*International Conference on Machine Translation of Languages and Applied Language Analysis*, National Physical Laboratory, Teddington, UK, 5-8 September 1961]

SYNTAX IN UNIVERSAL TRANSLATION

by

ITIROO SAKAI
(The First Research Centre of the
Defence Agency, 13, Mita, Meguro, Tokyo, Japan.)

1. INTRODUCTION.

EXPERIMENT is now underway on sentence to sentence translation from any language into any desired language. The machine used is a general purpose ferrite core parametron computer with magnetic core internal storage A magnetic drum and three magnetic tape handlers are provided as external storage media. A logical algebra has been introduced for analyzing and synthesizing text, so that the procedure be applicable to any language of any syntactical nature. The reference data of grammar are a set of fundamental rules prepared in a definite form, which are superposed digit by digit onto the text. The translation is carried out via machine language, a decoded version of the input text.

2. SEMANTIC UNIT, su.

Any symbol sequence shall be called a semantic unit or su. An su is a product of two su-s, if it can be split into two su-s, su' and su", so that the meaning and the syntactic function be derived from those of su' and su". We will express the relation by writing su = su' & su". The Symbol "&" is omitted whenever its meaning is clearly understood.

An su is an elementary semantic unit or esu, if it can no longer be split into shorter su-s or its meaning and syntactic function cannot be derived from its shorter components. Practically, the entries of an input dictionary are esu-s.

3. PART OF SPEECH, ps.

All the su-s have their own syntactic functions, which we will call parts of speech, or ps. The ps of an su, ps(su), for our operational purpose are defined so as to meet the following requirements.

(1) ps(su) = 0, if the su is not meaningful.

(2) ps(su') = ps(su") with context C, if both su' and su" display a syntactically identical role with respect to C.

(3) ps(su') = ps(su") with context set S, if ps(su') = ps(su") with all the contexts that are members of the set S.

(4) ps(su') = ps(su"), if ps(su') = ps(su") with S and if S contains all the contexts to be handled.

(5) ps(su) = ps(su') + ps(su") with context set S, if a set S = S' + S" exists, such that

ps(su) = ps(su') with S',
ps(su) = ps(su") with S".

(6) ps(su) = ps(su') + ps(su"), if ps(su) = ps(su') + ps(su") with S and if S contains all the contexts to be handled.

(7) ps(su') E ps(su), if real or imaginary su" exists, such that ps(su) = ps(su') + ps(su").

(8) ps(su) = ps(su') & ps(su"), if su = su(l) & su(2), ps(su') = ps(su(l)), ps(su") = ps(su(2)).

The symbol "&" is omitted whenever its meaning is clearly understood.

4. OPERATION RULES.

The su-s of a language can be classified after their ps-s. Let x, y, z, - - - be the variables representing ps-s. Their operation rules are assumed as follows.

(1) $x + y = y + x$

(2) $x + 0 = 0 + x = x$,

(3) $x + x = x$,

(4) $(x + y) + z = x + (y + z) = x + y + z$,

(5) $xy \neq yx$,

(6) $x \& 0 = 0 \& x = 0$,

(7) $(x + y)z = xz + yz$,

(8) $x(y + z) = xy + xz$,

(9) $(xy)z \neq x(yz)$,
(10) $xyz = (xy)z + x(yz)$,

(qq) $y = x/u/z$ if $u = xyz$,
$y = x/u/$ if $u = xy$,
$y = /u/z$ if $u = yz$.

5. ELEMENTARY PART OF SPEECH, eps.

A ps, x, is sometimes differentiated by its contexts into a sum of component ps-s. A component ps can or cannot be further differentiated. If possible, repeat the procedure in a similar way. We shall have a ps expanded in the form $x = e(l) + e(2) + - - - +e(r)=\sum'e(1)$

where e(i)-s are ps-s that can no longer be differentiated. They shall be called elementary parts of speech or eps-s. Take all the su-s in the input dictionary and expand their ps-s into eps-s. Let S be the set of all the eps-s thus obtained and let e(i) and e(j) be arbitrary eps-s of the set. The product e(i)e(j) is also a ps and can be expanded into eps-s. If a new kind of eps appears, it is added to the set S. Repeat the procedure until at last no more new kind of eps is obtained. The process is infinite, because otherwise the language itself will be of little practical use. The set thus obtained is the complete eps set of the language. The number of different eps-s in the complete set is determined by the language and the required quality of translation.

6. EXPANSION COEFFICIENT.

6.1. Let us consider a set of coefficients x(i), such that $x = \sum' u(i) = \sum x(i)u(i)$, where x(i) = 1 is u(i) E x,
x(i) = 0 if otherwise.
If the set u(i) is the complete eps set of the language, any ps is represented by $x = \sum' e(i) = \sum x(i)e(i)$.

6.2 Put $x = \sum x(i)e(i)$, $y = \sum y(j)e(j)$, $z = \sum z(k)e(k)$, and we have the following properties.

(1) If x + y = z, then x E z, y E z.
If e(i) E x, then x(i) = 1, e(i) E z;
if e(j) Ey, then y(j) = 1, e(j) Ez.
Therefore, by writing x(i) + y(j) = z(k), we have
0 + 0 = 0, 0 + 1 = 1 + 0 = 1 + 1 = 1.

(2) If xy = z, then $z = \sum x(i)e(i) \mathbin{\&} \sum y(j)e(j)$,
$= \sum\sum z(i,j)e(i)e(j)$.
e(i)e(j) E z, if and only if e(i) E x and e(j) E y.
Therefore, the expression x(i)y(j)= z(i,j), we have 0 x 0 = 0 x 1 = 1 x 0 = 0, 1 x 1 = 1.

(3) Expand the product in the form $e(i)e(j) = \sum a(i,j,k),e(k)$.
Writing $z = xy = \sum\sum\sum z(i,j)a(i,j,k)e(k) = \sum z(k)e(k)$, we have e(k) E z, if and only if e(i)e(j) Ez,
e(k) E e(i)e(j).
Therefore, in the expression z(i,j)a(i,j,k) = z(k), we have
0 x 0 = 0 x l = l x 0 = 0, 1 x 1 = 1.

6.3. The results above and the operation rules show that the product of any two ps-s are given by combining eps operations and that the grammatical rules may be prepared as a table of eps products. Applicable products are selected in reference to the expansion coefficients of the given ps-s. Thanks to the Boolean properties of the coefficients, we can imagine a ps product device in the form of a matrix. Each row corresponds to an eps and so does each column. The incoming ps, x, actuates some of the rows according to its expansion coefficients x(i) and y does similarly the column according to y(j). The cross point of the i-th row and the j-th column corresponds to x(i,j) which is 1 if and only if x(i) = y(j) = 1. The actuated cross points immediately call, up the ps code of e(i)e(j) in terms of their expansion coefficients a(i,j,k).

7. SYNTACTICAL STRUCTURE AND ITS TRANSFORMATION.

7.1. The syntactical structure of input text is determined by making all the possible combinations of su-s. The longest meaningful su thus obtained is the valid longest su, a product of two component su-s. If a component su is not an esu, it is again a product of two components. These su-s are really components of the longest meaningful su, and shall be called "valid su-s". The other su-s, that have been formed in the course of analysis, are "invalid" in the light of their context. The ps-s of these su-s are peculiar to the input language, and shall be called input parts of speech or ips. The ips combination u & v = w has its counterpart u' & v' = w' expressed in terms of machine language parts of speech or mps. The correspondence is given in the entries of input product table or ipt.

A proposition in our machine language has names and a verb. The names represent the participants and their relation is given by the verb. Any part of a proposition may have its modifier.

The mps-s of valid su-s are determined by looking up ipt again, picking up applicable entries in reference to their ips parts. The ipt carries marks of word order, which is 1 if the word order is transposed, and 0 otherwise. The text in machine language is then transformed into input language in a similar way. The opt, output product table, has also two parts, the one in machine language parts of speech and the other in output parts of speech or ops.

7.2. The structure transformation is illustrated by a simple example. It is in Japanese pronounced as if it is written in the international phonetic alphabet. The input text "joi hon' o jome" is dissolved into esu-s and numbered as

```
                jo - i     hon'    o     jom   -   e .
               (1)  2)     (3)    (4)    (5)      (6)
```

The input dictionary gives their meanings in machine codes as well as their ps-s.

```
ps(l) = ps(jo.) = adjectival stem = adj-st,
ps(2) = ps(-i)  = suffix to adj-st, yielding (/n/n)  or an
                         adjectival predicate
                = adj-st/ (/n/n) + p/.
ps (3) = ps (hon') = noun = n,
ps(4) = ps(o)   = bond representing the role of participant(s)
             /          it follows
      = b,
ps(5) = ps(jom-) = verbal stem = v-st,
ps(6) = ps(-e)   = suffix to v-st, yielding a verbal predicate
      = v-st/p/.
```

TABLE 1.

Simplified Eps Product Table, Japanese.

	n	p	b	x
n	n	0	x	0
p	0	0	x	0
b	0	0	0	0
x	0	p	x	0

TABLE 2.

Possible Su-s in ((12)(56)).

(12)	(13)	(14)	(16)
	(33)	(34)	(36)
		(44)	(46)
			(56)

Referring to *Table 1,* the ps-s of all the possible su-s in *Table 2* are obtained.

```
ps(l2) = adj-st & adj-st/(/n/n)  + p/ = /n/n + p,
ps(56) = v-st & v-st/p/ = p,
ps(13) = ps(12) & ps(33) = ((/n/n) + p)n = n,
ps(34) = ps(33) & ps(44) = nb = x,
ps(46) = ps(44) & ps(56) = bp = 0,
ps(14) =  ps(12)& ps(34) + ps(13) & ps(44)
       = ((/n/n) + p)x + nb = 0 + nb = x,
ps(36) = ps(33) & ps(46) + ps(34) & ps (56) = 0 + xp = p,
ps(16) = ps (12)& ps (36)+ ps (13)& ps (46) + ps (14) & ps (56)
      = ((/n/n) + p)p + 0 + xp = 0 + 0 + p=p.
```

The decoded form is obtained by the following operations.

```
ps(16) = p = xp = ps((14) (56)),
ps(14) = x = ps((13)(44)),
ps(13) = n =(/n/n)n = ps((12) (33)),
ps(12) = (/n/n) = (adj-st)(adj-st/(/n/n)/) = ps((11)(22)),
ps(11) = adj-st,
ps(22) = adj-st/(/n/n)/,
ps(33) = n,
ps(44) = b,
ps(56) = p = (v-st)(v-st/p/) = ps((55)(66)),
ps(55) = v-st,
```

ps(66) = v-st/p/).
It will be encoded into machine language.
(16) = verb(56) & participant(14),
(14) = bond(44) & substantive(13),
(13) = substantive(33) & modifier(12)

Referring to the output dictionary, from machine language into English for example, we have, since the modifier (12) corresponds to an English adjective of one word, the word order is (13) = ((12)(33)). The bond (44) means that the substantive (13) is a direct object of a predicate. The output product table gives the order (14) = ((44)(13)) and the output dictionary tells that it is represented by a space. The participant (14), being accusative by (44), follows (56), which is imperative by (66). Now, all put in English word order, (16) = ((56) (14)) =((56)((44)((12)(33)))).
Replacing semantic machine codes by English words,
(16) = read () good book(s).

8. REPRESENTATION AND APPLICATION OF SYNTACTICAL RULES

8.1. Most languages have detailed varieties of eps-s. An eps product table will include a tremendous number of entries if all the eps varieties are considered independent. Fortunately, they are classified into a handful of groups and the eps-s in a group are subclassified from various points of view, most of them being independent of each other.

An eps is represented by a series of figures. They are independent of each other. The figure 0 means that the viewpoint of this digit is of no concern. It is considered equal to any figure, if one digit is not enough for one point of view, two or more figures are combined: 00, 11, 12, 13, - - -, 21, 22, - - -, for example. When two representations are equal in the sense that the figure 0 is equal to all, they are in "agreement".

8.2. An elementary rule of grammar is given by an entry of eps product table in the form of a product. If the product is ambiguous as shown by u & v = w' + w", we establish two entries for w' and w".

TABLE 3.

Eps Product Table, schematized.

u	v	w	s	ord
1120	300700020	350100020	2	1
1120	300700020	350200020	2	1

8.3. Let us see an example of product operation. When a product b & c = d is wanted, with b = 1020 and c = 311721220 given, the product is indefinite before the operation: d = 000000000. By looking up the product table, the entries of *Table 3* are found to agree. The result (3) in *Table 4* is determined digit by digit.

TABLE 4.

Ps Operation on Product Table

(1) : b & c ≠ ? :	1020	311721220	000000000
(2) : u & v = w :	1120	300700020	350100020
(3) : b & c = d':	1120	311721220	350100020

A digit in (3) is equal to the corresponding digit in (2) if it is not a zero. If it is, the digit in (3) is equal to the corresponding digit of (1). The zeroes in the intermediate result d' are put equal to the corresponding digits of c when the value of s in the product table is 2. The application of the rule gives, as illustrated in *Table 5,* the result b = 1120, c = 311721220, d = 351121220, where the case of b is found nominative, which was indefinite 1020. The number of subject

TABLE 5

Transfer of Text Information

c :	311721220
d':	350100020
d :	351121220

still remains indefinite as indicated by the final 0 of b and d, which is to be determined by another entry of the table if an appropriate one is found available. The lower entry of *Table 3 i*s also in agreement in our present case. The procedure gives the result d - 351221220, another possibility of being "subjunctive".

9. REFERENCE DATA FOR TRANSLATION.

9.1 Input Dictionary, id. An entry of input dictionary has three parts: sp, sc and ips. Sp means spell in alphabetical notation. Sc is a numerical code for semantic specification of the entry. Ips is the part of speech in terms of the input language. A multiple meaning word is represented by as many entries. An entry that is the left half of an idiom has its ips "idiom'" and the right half "idiom"". The sc-s in these cases have no definite meanings but specifying an idiom.

9.2 Input Idiom Dictionary, idid. The idiom dictionary is identical to id in its structure except that sp is substituted by a couple of sc-s whose ips-s are idiom' and idiom". If an idiom is not yet completed, the ips of the entry is again idiom'. The two sc-s are shortened into a new sc.

9.3 Input Product Table, ipt. The table consists of three parts. The first is the input part u & v = w, the second is the corresponding machine language part u' *&* v' = w′ in terms of mps-s, and the third contains s and ord. If s is 0, no operation is made. If it is not 0, the operation runs as illustrated in *Table 5.* Similar operation is made with mps part, but usually in the reverse direction. See *Table 8.*

9.1 Ouptut Dictionary, od. The output dictionary is indexed by sc and mps. The information provided is ops and output spell. Ops is the part of speech in output language and the spell is given in terms of its location and its length. The entries are selected by coincidence of sc and agreement of mps. See *Table 10.*

9.4 Output Product Table, opt. Opt is a syntax transformation table from machine to output language. The mps part indicates the product u' & v' = w' and the output part gives the corresponding product in output language. Symbols to be inserted in the output language, bonds or word endings for example, are given before and after component ops-s u and v: ub before u, ub' after u, vb before v, vb' after v. The functions of s and ord' are similar to those of s and ord in ipt. See *Table 9.*

10. INTERMEDIATE DATA DURING TRANSLATION PROCESS.

10.1. The su storage, a magnetic drum of 24 x 20480 bits, is used to store all the intermediate data during the process. Its structure is roughly shown in *Table 13,* where a simple English text is translated into Japanese. The ps-s are in conventional notation for easier illustration. The table is modified so that the general method be understood at one glance. Practically, the spaces for analysis and synthesis are separated and the entries in the synthesis part carry the locations of corresponding entries of analysis. Sometimes, an entry of analysis corresponds to a few output counterparts. Proper output is chosen in virtue of syntactic and semantic indices in the ops.

10.2 An entry of su storage consists of the following sections. Brief interpretations are provided.

d: The entry number, represented by the location.
esu: If the entry is an esu, the value is 1; otherwise 0.
sc: If the entry is an esu, sc means semantic code. If not, the space is used for b and c, the entry location of the component su-s.
scd: Modifies the meaning of sc. If scd is 1, the sc is really an sc; if scd is 0, the sc means the location and length of input spell that was not found in id.
b,c; If the entry is a product of two su-s, the components are stored in the locations indicated by b and c.
h,k: The input order number of the esu-s standing at the beginning and the end of su. The su represented by the entry is a sequence of esu-s, from the h-th to the k-th.

ips: input part of speech given by id and ipt.
mps: Machine language part of speech out of ipt.
ops: Output part of speech given by od and opt.
ord, ord': If the input word order is interchanged in machine language, the ord is l, otherwise 0. Similarly, ord' is 1 or 0 according to the word order of machine and output languages.
val: If the entry is valid as an su of the longest valid su, it stores 1; if not 0. In synthesizing an output, figures 1 or 2 are added to the right end in a series. The series, when normalized to the left end, means the output word order.
sp: Location and length of output spell.
bc: Location and length of output spell to be inserted before sp.
bc': Location and length of output spell to be inserted after sp.

10.3 The contents of bc, sp and bc' of valid su-s are brought to the output storage. An entry of output storage has two cells, seq and sp. Seq is equal to val in the su storage with 1, 2 or 3 added to the right end to indicate that the entry is bc, sp or bc'. The output word order is determined by seq. The example in *Table 14* has an increased ambiguity for "solution" because of the modified presentation of su storage in *Table 13.*

TABLE 6.

We see that the solution is unique.
1 2 3 4 5 6 7

Possible su - s

(11)	(12)	(13)	(14)	(15)	(16)	(17)
	(22)	(23)	(24)	(25)	(26)	(27)
		(33)	(34)	(35)	(36)	(37)
			(44)	(45)	(46)	(47)
				(55)	(56)	(57)
					(66)	(67)
						(77)

expansion

$$(25) = (22)(35) + (23)(45) + (24)(55)$$

$$(h,k) = \sum(h,i)(j,k)$$

$$h \leq 1, \quad 1 + 1 = j \leq k$$

TABLE 7.

input dictionary		
spell	sc	ips
is	007	v(l)
is	007	v(4)
see	789	inf(l)
see	789	inf(2)
see	789	v(l)
see	789	v(2)
solution	83	n
that	8951	pn
that	8952	adj(pron)
that	8953	/n/s
that	8954	/(n/n/)/s
that	8955	/(n/n/)/v(l)
the	896	art
unique	93	adj
we	97	pn(l)

TABLE 8.

input product table							
u	v	w	s	u'	v'	w'	ord
a-n	v(l)	s	2	n(l)	v	s	1
art	n	a-n	2	n/n	n	n	1
inf(2)	n	inf(l)	1	v	n(4)	v	0
n	v(l)	s	2	n(l)	v	s	1
/n/s	s	n	0	n/s	s	n	0
/(n/n/)/s	s	n/n/	0	(n/n)/s	s	n/n	0
/(n/n/)/v(l)	v(1)	n/n/	0	(n/n)/v	v	n/n	0
pn(l)	v(l)	s	2	n(l)	v	s	1
v(2)	n	v(l)	1	v	n(4)	v	0
v(4)	adj	v(l)	1	v	v/v	v	0

TABLE 9.

output product table											
u'	v'	w'	ub	u	ub'	vb	v	vb'	w	s	ord'
n(1)	n/n	n(l)		n			/n/n		n	1	1
n/s	s	n(4)		n/p			p(4)	1408(1)	n	1	1
v	n(l)	s		p(4)	1425(1)		n	1401(2)	p(4)	1	1
v	n(4)	v		p(k-l)			n	1409(1)	p(k-l)	l	l
v	v/v	v		p(4)			n	1420(1)	p(4)	1	1
v	n(l)	s		p(k-l)	1415(1)		pn	1401(2)	p(k-l)	l	l

TABLE 10.

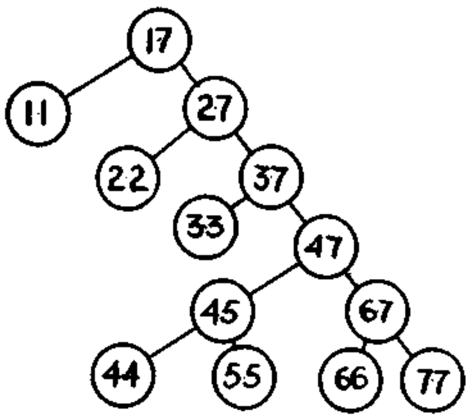
output dictionary				
			spell	
sc	mps	ops	address	(length)
007	v	p(4)	1153	(1)
789	v	p(k-l)	1176	(1)
83	n	n	1187	(7)
8951	pn	pn	1198	(1)
8952	n/n	adj(0)	1199	(1)
8953	n/s	n/s	2000	(1)
896	n/n	adj(0)	1201	(1)
93	v/v	n	1206	(3)
97	n(l)	pn	1237	(2)

TABLE 11.

output address	spell spell
1153	ar
1176	mi
1187	kota
1188	e/jo
1189	okai
1190	/joo
1191	eki/
1192	kaiz
1193	jo
1198	are.
1199	ano
1201	sono
1206	tada
1207	hito
1208	tsu
1237	ware
1238	ware
1401	ga/w
1402	a
1408	koto
1409	o
1415	ru
1420	de
1425	u
2000	xxxx

TABLE 12.

Structure of the Valid Su.



We see that the solution is unique
1 2 3 4 5 6 7

TABLE 13
Su Storage

d	h	k	b c	scd	esu	ips	mps	ops	val	ord	ord'	bc	sp	bc'
1	1	1	97	1	1	pn(1)	n(1)	pn	11				1237(2)	1401(2)
2	2	2	789	1	1	inf(1)								
3	2	2	789	1	1	inf(2)								
4	2	2	789	1	1	v(1)								
5	2	2	789	1	1	v(2)	v	p(k-1)	122				1176(1)	
6	1	2	1,4			s								
7	3	3	8951	1	1	pn								
8	3	3	8952	1	1	adj(pron)								
9	3	3	8953	1	1	/n/s	n/s	n/p	1212				2000(1)	
10	3	3	8954	1	1	/(n/n/)/s								
11	3	3	8955	1	1	/(n/n/)/v(1)								
12	2	3	3,7			inf(1)								
13	2	3	5,7			v(1)								
14	1	3	1,13			s								
15	4	4	896	1	1	art	n/n	adj(0)	121111				1201(1)	
16	5	5	83	1	1	n	n(1)	n	121112				1187(7)	
17	4	5	15,16			a-n	n(1)	n	12111	1	1			1401(2)
18	6	6	007	1	1	v(1)								
19	6	6	007	1	1	v(4)	v	p(4)	121122				1153(1)	
20	5	6	16,18			s								
21	4	6	17,18			s								
22	3	6	9,21			n								
23	3	6	10,21			n/n/								
24	2	6	3,22			inf(1)								
25	2	6	5,22			v(1)								
26	1	6	1,25			s								
27	7	7	93	1	1	adj	v/v	n	121121				1206(3)	1420(1)
28	6	7	19,27			v(1)	v	p(4)	12112	0	1			1425(1)
29	5	7	16,28			s								
30	4	7	17,28			s	s	p(4)	1211	1	1			1408(1)
31	3	7	9,30			n	n(4)	n	121	0	1			1409(1)
32	3	7	10,30			n/n/								
33	2	7	5,31			v(1)	v	p(k-1)	12	0	1			1415(1)
34	1	7	1,33			s	s		1	1	1			
35														
36														

TABLE 14.

Output

```
seq  112       113      1211112    1211122      121113
sp   1237(2)   1401(2)  1201(1)    1188(7)      1401(2)
      wareware  ga        sono      kotae          ga
               wa                  jookai         wa
                                   jooeki
                                   kaizjo

seq  1211212     1211213  1211222    121123     12113
sp   1206(3)     1420(1)  1153(1)    1425(1)    1408(1)
     tadahitotsu  de       ar         u          koto

seq  12122     1213      1222      123
sp   2000(1)   1409(1)   1176(1)   1415(1)
     xxxx      o          mi       ru
```

11. CONCLUSION

A common program for translating any language into any desired language has been developed by means of a sort of logical algebra. The method is universal including languages of greatly diverse syntax. Only fundamental rules are to be prepared and the applicable ones are chosen by digitwise agreement to be superposed onto the text. The properties of a language are programmed in an interpretive way, which can be revised easily whenever necessary.

ACKNOWLEDGMENT

The author appreciates the effort of the members of our centre N. Oobajasi, S. Nagahama and T. Kurasita, who devoted themselves to the verification of procedure.

APPENDIX

Abbreviations

eps:	elementary part of speech
esu:	elementary semantic unit
id:	input dictionary
idid:	input idiom dictionary
ips:	input part of speech
ipt:	input product table
mps:	part of speech in machine language
od:	output dictionary
ops:	output part of speech
opt:	output product table
ps:	part of speech
sc:	semantic code
sp:	spell
su:	semantic unit.