MULTILANGUAGE TRANSLATION OF NUMERALS BY PROLOG

Bengt Sigurd

INTRODUCTION

The present paper describes a program (Nummer.pro), which has been developed for PC (MSDOS) in PDC prolog (Prolog Development Center, DK-2605 Broendby). It translates numerals (English, Burmese, Georgian, French and Japanese) and is based on a general theory of numerals, which presumably can be applied to all numeral systems in the world, although with varying success - related to number of exceptions. Languages will show different types of deviations from the general model (cf below).

The general model accounting for the structure of numerals is formalized by the following rule:

Numeral --> ...(U G3)(U G2)(U G1)(U G0)

In this generative rule U is a unit numeral and G0,G1,G2,etc. are group numerals. The English numeral "two hundred" consists of the unit (U) numeral "two" and the group (G) word "hundred", which is G2 in the formula. There are no other constituents in this case. An expression within parentheses is called a group constituent. The meaning (value) of such a group constituent is the meaning of the unit word multiplied by the meaning of the group word, in this case 2*100=200, as expressed in the ordinary (decimal) mathematical system. A numeral may consist of one or several such group constituents. The numeral "three hundred and thirty three" consists of three group constituents (disregarding "and"):

(three hundred)(thir ty)(three)

Some of the irregularities hinted at above show up in this example. The unit numeral for 3 has different forms (the allomorphs three/thir) in its different occurrences and it seems to be more natural to treat thirty, and the numeral thirteen, as complexes not to be further analyzed synchronically (only etymologically). The English example also shows that there is no group word in the last constituent ("three"). This is the norm in languages, but there are languages such as Burmese, which have unit group words.

The group words in English are neat powers of ten (10), (10*10=100), (10*10*10=1000). We note that group words for 10000, 100000 are lacking in English - but not in the Burmese language, as will be shown. Group words with other values, such as 20 in French and many other languages imply multiplication by 20, etc. We note that the group words are ordered in increasing order right to left. There are obvious exceptions from this ordering rule in teens, e.g. "sixteen" if we assume that "teen" is the representative of the group word with the value 10. The teens are special. In German 32 would be "zwei-und-dreissig" which shows that the units may be placed before the tens even in higher numbers.

The "packing strategy" defined in Hurford(1975) means that the objects to be counted are packed in groups (containers) that are as big as possible. If there are 5200 objects, one does not say "fifty-two hundred", but "five thousand two hundred". If there are 62 objects, one does not say "sixty-twelve" in a decimal culture (but in a vigesimal such as the French). There are exceptions to this strategy, however. If years are counted, one says "fifteen hundred (and) twenty" instead of "one thousand five (hundred and) twenty". It is a moot question whether this violation of the packing strategy will be carried over into the twenty-first century. Will people say e.g. "twenty-one hundred (and) sixty-four" or "two thousand one hundred (and) sixty-four"? The Swedish Academy recommends the first solution for Swedish, but Swedes seem to hesitate. Teens are special and may be allowed to violate the packing strategy.

The meaning (value) of a string of constituents is also clear from our examples: the meaning of the whole expression is the sum of the values of the constituents. The derivation of the value of "three hundred and thirty-three" can then be illustrated by (3*100)+(3*10)+(3)=333. There is thus multiplication within the constituents, but addition between the constituents. Addition and multiplication are the standard operations in numeral systems, but subtraction has also been found, e.g. in Yoruba, where 105 may be expressed either as (20*5) + 5 or (5 from (10 from (20*6))). For detailed information about numerals see Hurford(1975) and Menninger (1958).

OVERVIEW

The program nummer.pro consists of modules for each language, and new modules can easily be added. Each language module understands the numerals of that language, i.e. gives the mathematical value if the operator writes check(lista) with a prefix, e.g. "jcheck" for Japanese, "gcheck" for Georgian, "bcheck" for Burmese. Each language module can also express a number in the language. This is done by writing say(number) with a prefix, e.g. fsay(89) which will return "quatre-vingt-dix-neuf".

Translation is done by deriving the mathematical value of the numeral of the source language by "check" and giving that number to "say" of the target language. The mathematical representation thus serves as an intermediate representation. A number of predicates are defined such as egtra(lista) which calls the English echeck and then Georgian say (gsay). These translation predicates are included in the language modules. The system is thus a multitranslation system for a special field. As the mathematical representations serve as a universal/international language in human communication such a multitranslation system for numerals may not seem useful, but there are certainly situations when it can be used. We note that the difficult task of representing the meaning of natural language is solved for numerals by using the mathematical representation and that the meaning (value) of the whole expression can nicely be computed from the meanings of the constituents. The modules for English and French are the most comprehensive programs.

UNDERSTANDING ENGLISH NUMERALS

Following is a Prolog program for English numerals up to 999999 (for certain languages less). It understands an English numeral, i.e. gives its decimal values if the numeral is written as an argument (a list within square brackets) of echeck, e.g. echeck([five,hundred,an,fifty,five]). (The word "and" is a built-in predicate, that is why "an" is used.) The program will return the value 555. Note that commas are needed within the words. The program can also go the other way and give numerals for decimal numbers. If we write esay(67), the program will return "sixty, seven". The PDC-Prolog gives an output with quotation marks around all list constituents, which is annoying but this is a matter of cosmetics. An improved print-out without quotation marks and commas is produced by prefixing "cosm" and writing e.g. cosmesay(67).

The program is based on the model presented above, but there are certain irregularities to observe. The teens have been taken as a special group. They are special as they denote 10 by "teen" and have the unit word before, not after as the word for 10 in the "ty"- numerals. The numeral "ten" is considered as a "ten" numeral in the program. The words "eleven","twelve", illustrate words which are best treated as unanalyzable although etymologically they include words equivalent to "one","two" and "leave" ("what is left over ten").

The consituent "num" handles hundreds with or without tens and units. This constituent is then used recursively before thousands in order to handle numerals as high as 999999 (although this particular Prolog takes time to handle such high numbers). We do not cover millions and billions in this exploratory program. The comments are meant to facilitate the understanding of the program.

```
A PROLOG PROGRAM UNDERSTANDING ENGLISH NUMERALS */

code=4500 /* allows memory organization */
domains
tal=real
lista=string*
predicates
num(tal,lista,lista)
enum(tal,lista,lista)
a(lista,lista)
uni(tal,lista,lista)
teen(tal,lista,lista)
deci(tal,lista,lista)
hund(tal,lista,lista)
tus(tal,lista,lista)
cent(tal,lista,lista)
mil(tal,lista,lista)
hy(lista,lista)
esay(tal)
cosmesay(tal) /* with cosmetic concatenation of morphs */
echeck(lista)
cosme(lista)
clauses
num(A,B,B1) :- uni(A,B,B1). /* e.g. two */
num(A,B,B1) :- teen(A,B,B1). /* e.g. twelve */
num(A,B,B1) :- deci(A,B,B1). /* e.g. twenty */
num(D,B,B3) :- deci(A,B,B1),hy(B1,B2),uni(C,B2,B3),D=A+C./* fifty-one */
num(A,B,B1) :- cent(A,B,B1). /* e.g. six hundred */
num(D,B,B3) :- cent(A,B,B1),a(B1,B2),uni(C,B2,B3),D=A+C. /* adds */
num(D,B,B3) :- cent(A,B,B1),a(B1,B2),teen(C,B2,B3),D=A+C.
num(D,B,B3) :- cent(A,B,B1),a(B1,B2),deci(C,B2,B3),D=A+C.
num(E,B,B5) :- cent(A,B,B1),a(B1,B2),deci(C,B2,B3),hy(B3,B4),uni(D,B4,B5
   ,E=A+C+D.
enum(A,B,B1) :- num(A,B,B1).
enum(A,B,B1) :- mil(A,B,B1).
enum(D,B,B3) :- mil(A,B,B1),a(B1,B2),uni(C,B2,B3),D=A+C.
enum(D,B,B3) :- mil(A,B,B1),a(B1,B2),teen(C,B2,B3),D=A+C.
enum(D,B,B3) :- mil(A,B,B1),a(B1,B2),deci(C,B2,B3),D=A+C.
enum(D,B,B2) :- mil(A,B,B1),num(C,B1,B2),D=A+C.
cent(D,B,B2) :- uni(A,B,B1),hund(C,B1,B2),D=A*C. /* multiplies */
mil(D,B,B2) :- num(A,B,B1),tus(C,B1,B2),D=A*C.
a([an|X],X). /* and occupied */
uni(1,[one|X],X).
uni(2,[two|X],X).
uni(3,[three|X],X).
uni(4,[four|X],X).
uni(5,[five|X],X).
uni(6,[six|X],X).
uni(7,[seven|X],X).
uni(8,[eight|X],X).
uni(9,[nine|X],X).
deci(10,[ten|X],X).
deci(20,[twenty|X],X).
deci(30,[thirty|X],X).
deci(40,[fourty|X],X).
deci(50,[fifty|X],X).
deci(60,[sixty|X],X).
deci(70,[seventy|X],X).
deci(80,[eighty|X],X).
deci(90,[ninety|X],X).
```

```
teen(11,[eleven|X],X).
teen(12,[twelve|X],X).
teen(13,[thirteen|X],X).
teen(14,[fourteen|X],X).
teen(15,[fifteen|X],X).
teen(16,[sixteen|X],X).
teen(17,[seventeen|X],X).
teen(18,[eighteen|X],X).
teen(19,[nineteen|X],X).
hund(100,[hundred|X],X).
tus(1000,[thousand|X],X).
echeck(X) :- enum(D,X,[]),write(D),nl. /* understands numeral X */
esay(X) :- enum(X,D,[]),write(D),nl. /* gives numeral for X */
cosmesay(X) :- enum(X,D,[]),cosme(D).
cosme(X) :- X=[H|T],T=[],write(H),nl.
cosme(X) :- X=[H|T],T=[H1|T1],concat(H,H1,H2),X1=[H2|T1],cosme(X1).
```

A PROLOG PROGRAM UNDERSTANDING BURMESE NUMERALS

The following is a representative set of Burmese numerals -

somewhat simplified and with commas between morphemes.

```
1 ta/ta,ku      11 hse,ta       10 hse/ta,hse    21 hna,hse,ta
2 hna           12 hse,hna      20 hna,hse       22 hna,hse,hna
3 thoun         13 hse,thoun    30 thoun,hse     33 thoun,hse,thoun
4 lei           14 hse,lei      40 lei,hse       44 lei,hse,lei
5 nga           15 hse,nga      50 nga,hse       55 nga,hse,nga
6 hcau          16 hse,hcau     60 hcau,hse      66 hcau,hse,hcau
7 hkun          17 hse,hkun     70 hkun,hse      77 hkun,hse,hkun
8 huy           18 hse,hyi      80 hyi,hse       88 hyi,hse,hyi
9 kou           19 hse,kou      90 kou,hse       99 kou,hse,ta
100 ta,ya                       101 ta,ya,ta
1000 ta,htaun                   654 hcau,ya,nga,hse,nga
10000 ta,thaun                 1965 ta,htaun,kou,ya,hcau,hse,nga
100000 ta,thein                6789 hcau,htaun,hkun,ya,hyi,hse,kou
```

The structure of the Burmese numerals is given by the rule:

bnumeral --> ..(U thein)(U thaun)(U htaun)(U ya)(U hse)(U ku)

The group words are powers of 10 in increasing order. The numeral for 1 can be with or without "ku", a group word for 1, roughly with the meaning 'pieces'. The unit word 1 may be optionally deleted before "hse" (10), in the numeral for 10; in the teens obligatorily. This cosmetic rule has not been implemented in this explorative program */

```
predicates
bnum(tal,lista,lista)
buni(tal,lista,lista)
bteen(tal,lista,lista)
bdeci(tal,lista,lista)
bhund(tal,lista,lista)
bcent(tal,lista,lista)
btus(tal,lista,lista)
bmil(tal,lista,lista)
btetus(tal,lista,lista)
bdmil(tal,lista,lista)
bsay(tal)
bcheck(lista)
betra(lista)
ebtra(lista)
clauses
bnum(A,B,B1) :- buni(A,B,B1).
bnum(A,B,B1) :- bdeci(A,B,B1).
bnum(D,B,B2) :- bdeci(A,B,B1),buni(C,B1,B2),D=A+C.
bnum(A,B,B1) :- bcent(A,B,B1).
bnum(D,B,B2) :- bcent(A,B,B1),buni(C,B1,B2),D=A+C.
bnum(D,B,B2) :- bcent(A,B,B1),bdeci(C,B1,B2),D=A+C.
bnum(E,B,B3) :- bcent(A,B,B1),bdeci(C,B1,B2),buni(D,B2,B3),E=A+C+D.
bnum(A,B,B1) :- bmil(A,B,B1).
```

```
bnum(D,B,B2) :- bmil(A,B,B1),buni(C,B1,B2),D=A+C.
bnum(D,B,B2) :- bmil(A,B,B1),bdeci(C,B1,B2),D=A+C.
bnum(E,B,B3) :- bmil(A,B,B1),bdeci(C,B1,B2),buni(D,B2,B3),E=A+C+D.
bnum(D,B,B2) :- bmil(A,B,B1),bcent(C,B1,B2),D=A+C.
bnum(E,B,B3) :- bmil(A,B,B1),bcent(C,B1,B2),bdeci(D,B2,B3),E=A+C+D.
bnum(E,B,B3) :- bmil(A,B,B1),bcent(C,B1,B2),buni(D,B2,B3),E=A+C+D.
bnum(F,B,B4) :- bmil(A,B,B1),bcent(C,B1,B2),bdeci(D,B2,B3),buni(E,B3,
   B4),F=A+C+D+E.
bnum(A,B,B1) :- bdmil(A,B,B1).
bnum(D,B,B2) :- bdmil(A,B,B1),bmil(C,B1,B2),D=A+C.
bnum(D,B,B2) :- bdmil(A,B,B1),bcent(C,B1,B2),D=A+C.
bnum(D,B,B2) :- bdmil(A,B,B1),bdeci(C,B1,B2),D=A+C.
bnum(D,B,B2) :- bdmil(A,B,B1),buni(C,B1,B2),D=A+C.
bnum(E,B,B3) :- bdmil(A,B,B1),bmil(C,B1,B2),bcent(D,B2,B3),E=A+C+D.
bnum(E,B,B3) :- bdmil(A,B,B1),bmil(C,B1,B2),bdeci(D,B2,B3),E=A+C+D.
bnum(E,B,B3) :- bdmil(A,B,B1),bmil(C,B1,B2),buni(D,B2,B3),E=A+C+D.
bnum(E,B,B3) :- bdmil(A,B,B1),bcent(C,B1,B2),bdeci(D,B2,B3),E=A+C+D.
bnum(E,B,B3) :- bdmil(A,B,B1),bcent(C,B1,B2),buni(D,B2,B3),E=A+C+D.
bnum(E,B,B3) :- bdmil(A,B,B1),bdeci(C,B1,B2),buni(D,B2,B3),E=A+C+D.
bnum(F,B,B4) :- bdmil(A,B,B1),bmil(C,B1,B2),bcent(D,B2,B3),bdeci(E,B3,
   B4),F=A+C+D+E.
bnum(F,B,B4) :- bdmil(A,B,B1),bmil(C,B1,B2),bcent(D,B2,B3),buni(E,B3,
   B4),F=A+C+D+E.
bnum(F,B,B4) :- bdmil(A,B,B1),bcent(C,B1,B2),bdeci(D,B2,B3),buni(E,B3,
   B4),F=A+C+D+E.
bnum(G,B,B5) :- bdmil(A,B,B1),bmil(C,B1,B2),bcent(D,B2,B3),bdeci(E,B3,
   B4),buni(F,B4,B5),G=A+C+D+E+F.
bdeci(D,B,B2) :- buni(A,B,B1),bteen(C,B1,B2),D=A*C.
bcent(D,B,B2) :- buni(A,B,B1),bhund(C,B1,B2),D=A*C.
bmil(D,B,B2) :- buni(A,B,B1),btus(C,B1,B2),D=A*C.
bdmil(D,B,B2) :- buni(A,B,B1),btetus(C,B1,B2),D=A*C.
buni(1,[ta|X],X).
buni(2,[hna|X],X).
buni(3,[thoun|X],X).
buni(4,[lei|X],X).
buni(5,[nga|X],X).
buni(6,[hcau|X],X).
buni(7,[hkun|X],X).
buni(8,[hyi|X],X).
buni(9,[kou|X],X).
bteen(10,[hse|X],X).
bhund(100,[ya|X],X).
btus(1000,[htaun|X],X).
btetus(10000,[thaun|X],X).
bcheck(X) :- bnum(A,X,[]),write(A),nl.
bsay(X) :- bnum(X,A,[]),write(A),nl.
betra(X) :- bnum(A,X,[]),esay(A). /* transl Burm X into Eng */
ebtra(X) :- enum(A,X,[]),bsay(A). /* transl Eng X into Burm */
```

TRANSLATING BETWEEN ENGLISH AND BURMESE

The English program presented above can give the mathematical representation when given an English numeral, or the English numeral when given the mathematical representation. Similarly, the Burmese program gives the mathematical representation, e.g. bcheck([lei,ya,nga]), which results in 405, or the Burmese numeral when given a number, e.g. bsay(4321), which gives "lei,htaun,thoun,ya,hna,hse,ta". The predicate betra(X), has been defined to handle translation between Burmese and English. It means: translate the Burmese numeral X into an English numeral. The predicate ebtra(X) translates from English into Burmese in a similar way.

GEORGIAN NUMERALS

Georgian numerals are illustrated by the following set.

1 erti	11 tertmeTi	10 ati	21 ocdaetri
2 ori	12 tormeTi	20 oci	22 ocdaori
3 sami	13 cameTi	30 ocvdaati	33 ocdacameTi
4 otxi	14 totxmeTi	40 ormoci	44 ormocidaotxi
5 xuti	15 txutmeTi	50 ormodaati	55 ormodatxutmeTi
6 ekvsi	16 tekvsmeTi	60 samoci	66 samocidaekvsi
7 Svidi	17 CvidmeTi	70 samocdaati	77 samocdaCvidmeTi
8 rva	18 tvrameTi	80 otxmoci	88 otxmocdarva
9 cxra	19 cxrameTi	90 otxmocdaati	99 otxmocdacxrameTi
100 asi	101 aserti	110 asati	121 asocdaerti
200 orasi	202 orasori	220 orasoci	240 orasormoci
300 samasi	355 samasormodatxutmeTi		
1000 atasi	2001 ori atas erti		
10000 ati atasi	1969 atascxraassamocdacxra		

The main structure of the Georgian numerals can be given by the following rule.

```
gnum --> ..(U X)(U asi)(U oci)(U ati)(U )
```

The Georgian system is a vigesimal system, a system where 20 assumes the value of the group word between 10 and 100. The formula above is in fact too compact and does not indicate where the additive marker (da) occurs.

The structure is shown by the following formula:

gnum --> .. (gnum as)(U oc) (da gtuni) -i

A Georgian assembles the items into twenties (oc) and the teens and units can be grouped together(gtuni). All numerals end in an -i, which disappears after another vowel (a, in rva,cxra). The teens are generally built by adding the unit word plus meTi "more" to "(a)t" (10). The word for fourty means 2 times 20, the word for fifty-five means 2 times 20 plus 15. Most of these facts are included in the following preliminary Prolog program. */

```
predicates
gnum(tal,lista,lista)
gtuni(tal,lista,lista)
gvingt(tal,lista,lista)
gcent(tal,lista,lista)
gmil(tal,lista,lista)
gsay(tal)
gcheck(lista)
getra(lista)
egtra(lista)
gbtra(lista)
bgtra(lista)
add(lista,lista)
end(lista,lista)
ghu(lista,lista)
gtu(lista,lista)
```

```
clauses
gnum(A,B,B2) :- gtuni(A,B,B1),end(B1,B2).
gnum(A,B,B2) :- gvingt(A,B,B1),end(B1,B2).
gnum(D,B,B4) :- gvingt(A,B,B1),add(B1,B2),gtuni(C,B2,B3),end(B3,B4),
   D=A+C.
gnum(A,B,B2) :- gcent(A,B,B1),end(B1,B2).
gnum(D,B,B3) :- gcent(A,B,B1),gvingt(C,B1,B2),end(B2,B3),D=A+C.
gnum(D,B,B3) :- gcent(A,B,B1),gtuni(C,B1,B2),end(B2,B3),D=A+C.
gnum(D,B,B5) :- gcent(A,B,B1),gvingt(C,B1,B2),add(B2,B3),gtuni(E,B3,
   B4),end(B4,B5),D=A+C+E.
gnum(A,B,B2) :- gmil(A,B,B1),end(B1,B2).
gnum(D,B,B3) :- gmil(A,B,B1),gcent(C,B1,B2),end(B2,B3),D=A+C.
gnum(D,B,B3) :- gmil(A,B,B1),gvingt(C,B1,B2),end(B2,B3),D=A+C.
gnum(D,B,B3) :- gmil(A,B,B1),gtuni(C,B1,B2),end(B2,B3),D=A+C.
gnum(E,B,B4) :- gmil(A,B,B1),gcent(C,B1,B2),gvingt(D,B2,B3),end(B3,B4)
   E=A+C+D.
gnum(E,B,B4) :- gmil(A,B,B1),gcent(C,B1,B2),gtuni(D,B2,B3),end(B3,B4),
   E=A+C+D.
gnum(E,B,B4) :- gmil(A,B,B1),gvingt(C,B1,B2),gtuni(D,B2,B3),end(B3,B4)
   E=A+C+D.
add([da|X],X).
end([i|X],X).
gvingt(20,[oc|X],X).
gvingt(40,[ormoc|X],X).
gvingt(60,[samoc|X],X).
gvingt(80,[otxmoc|X],X).
gtuni(1,[ert|X],X).
gtuni(2,[ori|X],X).
gtuni(3,[sam|X],X).
gtuni(4,[otx|X],X).
gtuni(5,[xut|X],X).
gtuni(6,[ekvs|X],X).
gtuni(7,[shvid|X],X).
gtuni(8,[rva|X],X).
gtuni(9,[cxra|X],X).
gtuni(10,[at|X],X).
gtuni(11,[tertmeT|X],X).
gtuni(12,[tormeT|X],X).
gtuni(13,[cameT|X],X).
gtuni(14,[totxmeT|X],X).
gtuni(15,[txutmeT|X],X).
gtuni(16,[tekvsmeT|X],X).
gtuni(17,[cCvidmeT|X],X).
gtuni(18,[tvrameT|X],X).
gtuni(19,[cxramet|X],X).
gcent(100,[at|X],X).
gcent(D,B,B2) :- gtuni(A,B,B1),ghu(B1,B2),D=A*100.
gmil(D,B,B2) :- gtuni(A,B,B1),gtu(B1,B2),D=A*1000.
ghu([as|X],X).
gtu([atas|X],X).
gcheck(X) :- gnum(A,X,[]),write(A),nl.
gsay(X) :- gnum(X,A,[]),write(A),nl.
getra(X) :- gnum(A,X,[]),esay(A). /* transl Geo X into Eng */
egtra(X) :- enum(A,X,[]),gsay(A). /* transl Eng X into Geo */
gbtra(X) :- gnum(A,X,[]),bsay(A).
bgtra(X) :- bnum(A,X,[]),gsay(A).
```

FRENCH NUMERALS

A representative set of French numerals is given below.

1 un/une	11 onze	10 dix
2 deux	12 douze	21 vingt et un
3 trois	13 treize	33 trente-trois
4 quatre	14 quatorze	44 quarante-quatre
5 cinque	15 quinze	55 cinquante-cinque
6 six	16 seize	66 soixante-six
7 sept	17 dix-sept	77 soixante-dix-sept
8 huit	18 dix-huit	88 quatre-vingt-huit
9 neuf	19 dix-neuf	99 quatre-vingt-dix-neuf
100 cent		1910 dix-neuf cent dix
1000 mille		2000 deux mille

French has some irregularities,e.g. inserting "et"(and) before un, but only in 21,31,41,51,61,71. A hyphen occurs within all numerals below one hundred. The system shows signs of the (Celtic) vigesimal counting. The following rule shows the main structure of French numerals:

fnum --> ..(U mille)(U cent)(ante)-(fteen)

It is clear that the "ante" words are of two kinds, those based on vingt (multiplied with 4 in the word for 80) and the others. Note that although the word "soixante" is not based on 20, it takes teens to denote the numerals for 70-79. Gender inflection is disregarded (e.g. "un:une"). The word for 100 ("cent") is plural ("cents") after plural numerals, but not when lower numerals follow. This has been implemented in the program where "cent" is treated separately also because "un" is deleted before "cent" (and "mille"). Most of the basic facts of French numerals are covered by the following Prolog program, where some of the irregularites stand out. */

```
predicates
fnum(tal,lista,lista)
cfnum(tal,lista,lista)
fun(tal,lista,lista)
funi(tal,lista,lista)
fteen(tal,lista,lista)
fvingt(tal,lista,lista)
etante(tal,lista,lista)
ante(tal,lista,lista)
fcent(tal,lista,lista)
fcents(tal,lista,lista)
funmil(tal,lista,lista)
fmil(tal,lista,lista)
ftus(lista,lista)
et(lista,lista)
fsay(tal)
cfsay(tal)
fcheck(lista)
fetra(lista)
eftra(lista)
fbtra(lista)
bftra(lista)
fgtra(lista)
gftra(lista)
clauses
fnum(A,B,B1) :- funi(A,B,B1).
fnum(A,B,B1) :- fteen(A,B,B1).
fnum(A,B,B1) :- ante(A,B,B1).
fnum(A,B,B1) :- fvingt(A,B,B1).
fnum(D,B,B3) :- etante(A,B,B1),et(B1,B2),fun(C,B2,B3),D=A+C.
fnum(71,B,B3) :- fvingt(60,B,B1),et(B1,B2),fteen(11,B2,B3).
fnum(D,B,B3) :- ante(A,B,B1),hy(B1,B2),funi(C,B2,B3),D=A+C.
fnum(D,B,B3) :- fvingt(A,B,B1),hy(B1,B2),funi(C,B2,B3),D=A+C.
fnum(D,B,B3) :- fvingt(A,B,B1),hy(B1,B2),fteen(C,B2,B3),D=A+C.
fnum(A,B,B1) :- fcent(A,B,B1).
fnum(C,B,B2) :- fcent(A,B,B1),funi(D,B1,B2),C=A+D.
fnum(C,B,B2) :- fcent(A,B,B1),fteen(D,B1,B2),C=A+D.
fnum(C,B,B2) :- fcent(A,B,B1),fvingt(D,B1,B2),C=A+D.
fnum(C,B,B2) :- fcent(A,B,B1),ante(D,B1,B2),C=A+D.
fnum(C,B,B4) :- fcent(A,B,B1),etante(D,B1,B2),et(B2,B3),fun(E,B3,B4),
   C=A+D+E.
fnum(C,B,B4) :- fcent(A,B,B1),ante(D,B1,B2),hy(B2,B3),funi(E,B3,B4),
   C=A+D+E.
fnum(C,B,B4) :- fcent(A,B,B1),fvingt(D,B1,B2),hy(B2,B3),funi(E,B3,B4),
   C=A+D+E.
fnum(C,B,B4) :- fcent(A,B,B1),fvingt(D,B1,B2),hy(B2,B3),fteen(E,B3,B4)
   C=A+D+E.
fnum(D,B,B2) :- funi(A,B,B1),fcents(C,B1,B2),D=A*C.
fnum(D,B,B3) :- funi(A,B,B1),fcent(C,B1,B2),funi(E,B2,B3),D=A*C+E.
fnum(D,B,B3) :- funi(A,B,B1),fcent(C,B1,B2),fteen(E,B2,B3),D=A*C+E.
fnum(D,B,B3) :- funi(A,B,B1),fcent(C,B1,B2),ante(E,B2,B3),D=A*C+E.
fnum(D,B,B3) :- funi(A,B,B1),fcent(C,B1,B2),fvingt(E,B2,B3),D=A*C+E.
fnum(D,B,B5) :- funi(A,B,B1),fcent(C,B1,B2),ante(E,B2,B3),hy(B3,B4),
   funi(F,B4,B5),D=A*C+E+F.
fnum(D,B,B5) :- funi(A,B,B1),fcent(C,B1,B2),etante(E,B2,B3),et(B3,B4),
   fun(F,B4,B5),D=A*C+E+F.
fnum(D,B,B5) :- funi(A,B,B1),fcent(C,B1,B2),fvingt(E,B2,B3),hy(B3,B4),
   funi(F,B4,B5),D=A*C+E+F.
fnum(D,B,B5) :- funi(A,B,B1),fcent(C,B1,B2),fvingt(E,B2,B3),hy(B3,B4),
   fteen(F,B4,B5),D=A*C+E+F.
```

```
cfnum(A,B,B1) :- fnum(A,B,B1).
cfnum(A,B,B1) :- funmil(A,B,B1).
cfnum(D,B,B2) :- funmil(A,B,B1),fnum(C,B1,B2),D=A+C.
cfnum(A,B,B1) :- fmil(A,B,B1).
cfnum(D,B,B2) :- fmil(A,B,B1),fnum(C,B1,B2),D=A+C.
hy([_|X],X).
et([et|X],X).
fun(1,[un|X],X).
funi(1,[un|X],X).
funi(2,[deux|X],X).
funi(3,[trois|X],X).
funi(4,[quatre|X],X).
funi(5,[cinque|X],X).
funi(6,[six|X],X).
funi(7,[sept|X],X).
funi(8,[huity|X],X).
funi(9,[neuf|X],X).
fteen(10,[dix|X],X).
fteen(11,[onze|X],X).
fteen(12,[douce|X],X).
fteen(13,[treize|X],X).
fteen(14,[quatorze|X],X).
fteen(15,[quinze|X],X).
fteen(16,[seize|X],X).
fteen(17,[dix_sept|X],X).
fteen(18,[dix_huit|X],X).
fteen(19,[dix_neuf|X],X).
ante(20,[vingt|X],X).
ante(30,[trente|X],X).
ante(40,[quarante|X],X).
ante(50,[cinquante|X],X).
etante(20,[vingt|X],X).
etante(30,[trente|X],X).
etante(40,[quarante|X],X).
etante(50,[cinquante|X],X).
etante(60,[soixante|X],X).
fvingt(60,[soixante|X],X).
fvingt(80,[quatre_vingt|X],X).
fcent(100,[cent|X],X).
fcents(100,[cents|X],X).
ftus([mille|X],X).
fmil(C,B,B2) :- fnum(A,B,B1),ftus(B1,B2),C=A*1000.
funmil(1000,B,B1) :- ftus(B,B1).
fsay(X) :- cfnum(X,Y,[]),write(Y),nl.
cfsay(X) :- cfnum(X,Y,[]),cosme(Y).
fcheck(Y) :- cfnum(X,Y,[]),write(X),nl.
fetra(X) :- cfnum(A,X,[]),esay(A),nl.
eftra(X) :- enum(A,X,[]),fsay(A),nl.
fbtra(X) :- cfnum(A,X,[]),bsay(A),nl.
bftra(X) :- bnum(A,X,[]),fsay(A),nl.
fgtra(X) :- cfnum(A,X,[]),gsay(A),nl.
gftra(X) :- gnum(A,X,[]),fsay(A),nl.
```

JAPANESE NUMERALS

The following is a representative set of Japanese numerals

1 ichi	11 ju_ichi	10 ju
2 ni	12 ju_ni	22 ni_ju_ni
3 san	13 ju_san	33 san_ju_san
4 yon (shi)	14 ju_yon	44 yon_ju_yon
5 go	15 ju_go	55 go_ju_go
6 roku	16 ju_roku	66 roku_ju_roku
7 nana (shichi)	17 ju_nana	77 nana_ju_nana
8 hachi	18 ju_hachi	88 hachi_ju_hachi
9 ku (kyu)	19 ju_ku	99 ku_ju_ku

100 hyaku
150 hyaku_go_ju
200 ni_hyaku
300 san_hyaku
1000 sen
2000 ni_sen
10 000 ichi_man

The system is fairly regular if we disregard phonological variation and some other complications, which imply e.g. that san_hyaku is rendered by san_byaku, the word for 600 is roppyaku, that for 800, happyaku, 900, kyuhyaku, 3000, sanzen, 8000, hassen, 9000, kyusen. We also disregard the problem of classifiers in Japanese (and Chinese). We suggest the following preliminary program */

```
predicates
jnum(tal,lista,lista)
juni(tal,lista,lista)
jten(lista,lista)
jhu(lista,lista)
jcent(tal,lista,lista)
jdec(tal,lista,lista)
jteen(tal,lista,lista)
jcheck(lista)
jsay(tal)
jbtra(lista)
bjtra(lista)
ejtra(lista)
jetra(lista)
fjtra(lista)
jftra(lista)
jgtra(lista)
gjtra(lista)
clauses
jnum(A,B,B1) :- juni(A,B,B1).
jnum(A,B,B1) :- jteen(A,B,B1).
jnum(A,B,B1) :- jdec(A,B,B1).
jnum(D,B,B2) :- jdec(A,B,B1),juni(C,B1,B2),D=A+C.
jnum(A,B,B1) :- jcent(A,B,B1).
jnum(D,B,B2) :- jcent(A,B,B1),juni(C,B1,B2),D=A+C.
jnum(D,B,B2) :- jcent(A,B,B1),jdec(C,B1,B2),D=A+C.
jnum(D,B,B3) :- jcent(A,B,B1),jdec(C,B1,B2),juni(E,B2,B3),D=A+C+E.
jteen(D,B,B2) :- jten(B,B1),juni(C,B1,B2),D=10+C.
jdec(D,B,B2) :- juni(A,B,B1),jten(B1,B2),D=A*10.
jcent(D,B,B2) :- juni(A,B,B1),jhu(B1,B2),D=A*100.
juni(1,[ta|X],X).
juni(2,[ku|X],X).
juni(3,[san|X],X).
juni(4,[yon|X],X).
juni(5,[go|X],X).
juni(6,[roku|X],X).
juni(7,[nana|X],X).
juni(8,[hachi|X],X).
juni(9,[ku|X],X).
jten([ju|X],X).
jhu([hyaku|X],X).
```

```
jcheck(X) :- jnum(A,X,[]),write(A),nl.
jsay(X) :- jnum(X,Y,[]),write(Y),nl.
jetra(X) :- jnum(A,X,[]),esay(A). /* transl Jap X into Eng */
ejtra(X) :- enum(A,X,[]),jsay(A).
jbtra(X) :- jnum(A,X,[]),bsay(A).
bjtra(X) :- bnum(A,X,[]),jsay(A).
jgtra(X) :- jnum(A,X,[]),gsay(A).
gjtra(X) :- gnum(A,X,[]),jsay(A).
fjtra(X) :- cfnum(A,X,[]),jsay(A).
jftra(X) :- jnum(A,X,[]),fsay(A).
```

CONCLUSION: TYPOLOGICAL FEATURES OF NUMERALS

Studies of numerals and computer programs which understand or translate numerals show that numeral systems are variants of certain themes (parameters). The lexical material (the morphemes) varies, of course, but in addition we can see variation in the following features.

1. Deletion of 1 (uni-deletion) before group words, e.g. before hundred, optionally; mostly obligatorily before low group words.
2. Insertion of conjuction (and-,et-,&-insertion), between high and low group words.
3. Order between low group constituents, e.g. units before tens.
4. Group word values, typically 10, 20 (as in French, Danish, Welsh, Yoruba, Mixtec), but also 5 (as in Ainu, Bantu, Khmer).
5. Group word value relations, e.g. powers of 10, multiples of 5 or 20. The series of group words is often irregular and broken, and e.g. a word for 10 000 and higher may be lacking.
6. Combination of morphemes into words or word groups and hyphenation. In languages with old orthography such as French the rules for the occurrence of hyphens are very rigid.

These features show what is generally to be taken into account when designing a computer program understanding or translating between languages. The programs above may be used as modules in a more comprehensive translating system. Other languages can easily be added. Prolog programs can reflect the linguistic intutions about numeral systems well, but they are certainly not the fastest programs which can be implemented for handling numerals. The programs presented need cosmetic improvement, but they should be good illustrations of one Prolog approach to numerals.

REFERENCES

Corstius, H.B. (ed), Grammars for number names. Foundations of Language series Vol 7, Dordrecht(1968:Reidel)
Hurford, J.R.The linguistic theory of numerals, Cambridge (1975:Cambridge University Press)
Menninger, K. Zahlwort und ziffer. Ein Kulturgeschichte der Zahl, Goettingen (1958:Vandenhoeck & Ruprecht)
Sigurd, B. From mathematical into linguistic representations - and vice versa. Papers from the institute of linguistics, Stockholm (PILUS),15 (1972)
" " Round numbers. Language in Society,1988 (to appear)