HUGO BRANDT CORSTIUS

AUTOMATIC TRANSLATION BETWEEN NUMBER NAMES*

1. INTRODUCTION

One of the problems in mechanical translation is the occurrence of compounds without spaces in languages as German and Dutch; these cannot be put in a dictionary as any language user can produce them without limit. Little has been done on this problem. In the pioneer days of mechanical translation, E. Reifler indicated [1] how the syntactic analysis of the German compound can be done by computer. The semantic problems are almost unsurpassable. These semantic problems are quite simple for one large class of compound words: the number names. From a small number of components, around thirty, most languages can construct millions of cardinal number names. For five languages, Dutch, German, French, English and Chinese, the rules governing the construction of number names were programmed. The translation from the decimal representation into the word representation is effected by the ALGOL procedure *write number name* (Section 2).

Conversely the procedure *read number name* gives the translation from word form to digital form (Section 3). Combination of these procedures then gives the mutual translation of number names (Section 4).

2. SYNTHESIS OF THE NUMBER NAME

Recently, a number of generative grammars, most of them context-free, have been published for number names in different languages [2]. Whatever the merits of these grammars, they all possess one shortcoming: the names generated have no connection with their meaning. And this while the determination of the meaning of number names is so simple. Except for extra connotations of some smaller numbers the meaning of a number name is fully given by the decimal representation of the number referred to. Therefore we have adapted the generative grammars to make possible a translation from decimal to word representation. Moreover, by treating Dutch, German, French and English in one program, a comparative grammar of the number names in these languages is found. The set of Chinese number names is generated on its own.

* This is chapter 2 of the author's doctoral thesis *Exercises in Computational Linguistics.*

Brandt Corstius (ed.), Grammars for Number Names, 103–123.

In the four Western languages, the number is partitioned from the back in groups of three digits. The procedure *next three digits* (i) (p. 115) gives the word representation of one such group of three digits, followed by the appropriate power of ten, and then calls itself with lowered value of i, until the end of the number is reached. The procedure makes use of three other procedures:

from 1 up to 100(j, k)	producing $10\times j+k$	(p. 116)
hundredfold (j, k)	producing $(10\times j+k)\times 100$	(p. 117)
thousand to the power (k)	producing 10^{3k}	(p. 117)

In some languages, it is possible under certain conditions to write a number like 1200 in two ways: as 12×100 and as $1000+200$. The program investigates whether this situation is present, and, if so, gives both possibilities. (Theoretically one could perhaps write a number like 1200001200 in these languages in four ways. We have used the 'overlapping mode' or the 'non-overlapping mode' consistently in the whole number name.) For many special measures as e.g. the declension of the German powers of ten and the French numbers between 70 and 99, we refer to the program. The French word for 10^9 is given half of the time as *billion*, and the other half as *milliard.* For English we follow the British system of naming 10^9 'milliard' and 10^{12} 'billion'. In the American system these names are 'billion' and 'trillion'. The morphemes used can be found in the first four columns of procedure m (on p. 114). For each of the four languages, about 33 morphemes are provided, of which some could be omitted, but at the cost of program complications. The Chinese number name can be generated by a context-free grammar but Brainerd has shown [3] that it is better to use a class of deletion transformations which delete one component in specified contexts. These deletion transformations are readily programmable in ALGOL 60. The transformations in connection with the morpheme *ling* ('zero') are simply written as procedure D (*condition for context around ling, deletend*), where, each time the Boolean *condition for context around ling* is true, the component at place *deletend* is deleted. For details we refer again to the ALGOL program (pp. 118, 119, 120). Besides the seven obligatory transformations, there are five optional transformations which give alternative forms. The necessary component morphemes of the Chinese number names can be seen in the last column of procedure m (on p. 114).

3. ANALYSIS OF THE NUMBER NAME

On the basis of the synthesis in the preceding section, an analysis could be given. This method, however, is:

(1) unnecessary, because many of the special measures in the procedure *write number name* are taken to ensure a correct writing of the number name, but have no informational value. In a situation where we want to read a number name, a less detailed grammar is good enough.

(2) undesirable because we also want to read in alternative forms which are not completely correct, as they often occur for larger numbers, which are rarely written in word form. We can e.g. neglect spacing and capitalisation, and accept rare forms like *six-and-twenty*.

A second method for analysis is the Reifler procedure which consists, briefly, in taking the morpheme in the component dictionary which coincides with the longest left end of the word to be dissected. If, in repeating this strategy, no component can be found at some stage, the last taken decision is revoked and a shorter component chosen. This full analysis is not necessary in our case because, only in two cases (one of which is caused by the accidentally chosen transcription of the Chinese words) is a wrong analysis found if we always choose the longest possible component fitting the left end of the name to be analyzed. To simplify the program, components of only 1, 2, 3 or 4 letters are used. It appears that with some tricks, the first four letters give enough discrimination between the components. Three numbers are associated with each component. The first number indicates the language in which the component occurs (as soon as the machine knows that it is in a certain language, only the relevant part of the dictionary is searched for the other components). The second number indicates the number of letters the full component possesses (of which, as has been said, only four were used for identification), and the third gives the meaning of the component. These meanings are coded as follows:

code	meaning	example
0	meaningless	*et*, *ling*, the *s* in sech*s* and cent*s*
1	1	*ein*, *i*
⋮	≡ code	numbers below 20
16	16	*seize*
20	10-fold suffix	*tig*, *zig*, *uante*, *ty*
21	10	*dix*
22	20	*twenty*
⋮	(code-20) × 10	tenfolds
24	40	*quarante*
30	connective between tenfolds and units	*en*, *und*

code	meaning	example
32	10^2	*hundred*
33	10^3	*mille*
⋮	$10^{code-30}$	powers of ten
42	10^{12}	*trillion*
51	even power of thousand suffix	*on, oen*
52	odd power of thousand suffix	*ard*
53	mil-	*milli, milj*
54	bil-	*billi, bilj*

For the analysis of the number names in five languages, a hundred components are necessary, less than for synthesis. This is because words like *veertien* and *veertig* are synthesized as a whole, but in analysis are broken down further (for analysis *veer* is fully synonymous with *vier*, which it is not in synthesis), and because some components occur in more than one language (usually with the same meaning: *acht*, *six*. Exception: *billion*). The procedure (pp. 108–112) reads the letters of the number name offered into the array *L*, thereby discarding some information, which it also discarded reading the list of component words. The four-, three-, two- and one-letter combinations are, in that order, compared with the entries in the component list. The meanings of the components found are stored in the array *M*. In the meantime, the tenfolds, the numbers between 10 and 20, and the powers of ten above 10^3 are combined. The notation used for Chinese morphemes makes it necessary to discrimate between *pa i*^{4} (8×10^4) and *pai* (100). The optional transformation O2 obliges us to see whether *ling* has just been read. Also, we have to be careful not to read the English *one* in the German *Billionen*. In the languages which put the units before the tenfold (connected by *and*) for the numbers below 100, this order is reversed. Except for the ambiguous *billion*, the series of meanings in *M* is now independent of the input language. From back to front, the elements of *M* are then converted into digits in the array *N*, which will contain the number in decimal representation. To this end, a counter *pos* is kept, whose initial value is 0. The powers of ten above 10^3 give pos a new absolute value, and the lower powers of ten enlarge *pos* (additively). When we have reached the first element of *M*, the digital representation is finished and *number of digits* has acquired its value.

The procedure *read number name* reads all words produced by the procedure *write number name*, plus many incorrect forms. Spaces, capitalization, hyphens and apostrophes are neglected.

4. TRANSLATION OF NUMBER NAMES

It is clear how the two procedures *write number name* and *read number name* make mutual translation in the five languages possible. In the case where input language and output language are the same this translation can be interpreted as: 'Give the correct version of the number name plus all its alternative forms'.

The program reproduced here gives, for every number name read, its translation in the five languages. In the output, the input is reproduced with underlinings, including the component dictionary. The maximal number processed is $10^{15}-1$ but extension is simple (for Dutch the synthesis up to $10^{66}-1$ is done in [4]). In such an extension the difference between the American and the English system of denominations above one million becomes more pronounced.

In the case of number name translation between several languages, the use of the decimal representation as an intermediate language seems obvious. The use of an intermediate language for the mutual translation of several languages in general may, however, very well be impossible or undesirable.

Apart from the actual application discussed, it is hoped that the published program may convince linguists that general programming languages, such as ALGOL 60, offer a means for concise, controllable description of complex situations.

Mathematical Centre, Amsterdam

BIBLIOGRAPHY

[1] E. Reifler, 'Mechanical Determination of the Constituents of German Substantive Compounds', *MT* **2** (1955), 3–14.

[2] We refer in the first place to the articles in this volume. For French we refer to: R.P.G. de Rijk, 'Une grammaire "context-free" pour la génération mécanique des noms de nombres français', in: Braffort and F. van Schepen, *Automation in language translation etc.*, Euratom CID, Brussels, 1967. For Chinese we used [3].

[3] B. Brainerd, 'Two Grammars for Chinese Number Names', *Canadian Journal of Linguistics* **12** (1966), 33–51.

[4] H. Brandt Corstius, 'Automatic Translation of Numbers into Dutch', *Foundations of Language* **1** (1965), 59–62.

```
begin comment mutual translation of number names;
  integer number of digits, lang, dutch, german, french, english, chinese;
  Boolean overlap, possible overlap, o1, f, one;
  integer array W[1:108, 1:4], A[0:5, 0:1], N[1:16];
  comment W contains the components with their meanings. A gives the
  limits for each language in the list W. N stores the digits of the number
  translated;

  integer procedure letter;
  comment This input procedure reads the next letter of the input word. It
  neglects spaces, hyphens, apostrophes, capitalization and the letters q and
  m. It recodes the letter into the code a=1,...z=26. On reading a question
  mark the program is terminated. The input text is repeated, underlined, in
  the output;
  begin integer h;
    h:=RESYM; if h≠119 then SYM(126); SYM(h); if h=122 then EXIT;
    letter:=if h=93 ∨ h=120 ∨ h=65 ∨ h=53 ∨ h=26 ∨ h=49 ∨ h=22 then
    letter else if h=4 then 22 else if h>36 then h−36 else h−9
  end;

  procedure read number name;
  comment This procedure translates the number named by the input word
  into the decimal representation. lang gets the value of the input language;
  begin integer h, j, iL, iLmax, iM, iMmax, iN, w1, w2, w3, w4, m, b, pos, ling;
    integer array L[1:170], M[0:40];
    comment L stores the letters of the input word, M the meanings of the
    found components;

    procedure search(n, m); value n; integer n, m;
    comment The word n is looked up in W. If found m becomes equal to the
    index of the corresponding morfeme, else m is made negative. The
    language to which the found morfeme belongs is kept for future searches;
```

```
begin integer iW;
   for iW:=A[lang, 0] step 1 until A[lang, 1], 1 step 1 until A[0, 0]−1
   do
   begin if W[iW, 1]=n then
      begin if lang=0 then lang:=W[iW, 2]; m:=iW; goto FOUND
      end a component is found
   end of search in relevant part of W;
   m:=−1;
FOUND:
end search;

SL: for iL:=1 step 1 until 167 do
   begin h:=letter; if h=83 then
      begin if iL=1 then goto SL; iLmax:=iL−1; goto Lfilled
      end the letters of the input word are now stored in L;
      L[iL]:=h
   end;
   goto SL;
Lfilled: lang:=M[0]:=ling:=0;
   L[iLmax+1]:=L[iLmax+2]:=L[iLmax+3]:=26; iL:=iM:=1;
SM: if iL>iLmax then goto Mfilled; w1:=L[iL];
   w2:=w1×26+L[iL+1]; w3:=w2×26+L[iL+2];
   w4:=w3×26+L[iL+3]; search(w4, m); if m<0 then
   begin search(w3, m); if m<0 then
      begin search(w2, m); if m<0 then
         begin search(w1, m); if m<0 then goto SL end
      end
   end the first 4, 3, 2 and 1 letters of the input word have been looked up in W;
   iL:=iL+W[m, 3]; b:=W[m, 4];
   if lang=chinese then
   begin if b=0 then ling:=1 else ling:=ling+1;
      if b=32 ∧ L[iL]=22 then
      begin b:=8; iL:=iL−1 end pa i
   end of special measures for Chinese;
```

```
if lang=english then
begin if b=1 ∧ (M[iM−1]=53 ∨ M[iM−1]=54) then
  begin lang:=0; b:=51; iL:=iL−1 end
end of special measures for English;
if b=0 then goto SM; if b=20 then
begin M[iM−1]:=M[iM−1]+20; goto SM
end forming of tenfolds;
if b=10 then
begin if lang=chinese then
  begin if M[iM−1]<10 ∧ iM≠1 then
    begin M[iM−1]:=20+M[iM−1]; goto SM end
   else b:=21
  end Chinese tenfolds;
  if lang=french then
  begin if M[iM−1]=26 ∨ M[iM−1]=28 then
    begin M[iM−1]:=M[iM−1]+1; goto SM end
  end French 70 and 90
  else
  begin if M[iM−1]<10 ∧ M[iM−1]>2 then
    begin M[iM−1]:=M[iM−1]+10; goto SM end
  end forming of numbers between 10 and 20
end occurrence of 10;
if lang=french then
begin if b>6 ∧ b<10 then
  begin if M[iM−1]=10 then
    begin M[iM−1]:=10+b; goto SM end
  end French 17, 18, 19;
  if b=22 ∧ M[iM−1]=4 then
  begin M[iM−1]:=28; goto SM
  end French 80
end of special measures for French;
if b=51 ∨ b=52 then
begin if M[iM−1]=53 then
  begin M[iM−1]:=30+(if b=51 then 6 else 9); goto SM
  end million and milliard;
```

```
    if M[iM−1]=54 then
    begin M[iM−1]:=41; goto SM
    end The ambiguous billion is given meaning 41 until we know whether the
    input language is French or not
  end;
  if lang=german then
  begin if m=6 then goto SM
  end special measure for German sieben;
  M[iM]:=b; if iM=40 then goto SL; iM:=iM+1; goto SM;
Mfilled: iMmax:=iM−1; if iMmax=0 then goto SL;
  if lang=chinese ∧ ling≠2 ∧ M[iMmax]<10 ∧ M[iMmax−1]≠10
  then
  begin if M[iMmax−1]=32 then M[iMmax]:=20+M[iMmax] else if
    M[iMmax−1]=33 ∨ M[iMmax−1]=34 then
    begin iMmax:=iMmax+1; M[iMmax]:=M[iMmax−2]−1 end
  end The effect of optional transformation O2 is reversed;
  if lang=dutch ∨ lang=german ∨ lang=english then
  begin for iM:=2 step 1 until iMmax−1 do
    begin if M[iM]=30 then
      begin if M[iM−1]<10 ∧ M[iM+1]>21 ∧ M[iM+1]<30 then
        begin m:=M[iM−1]; M[iM−1]:=M[iM+1]; M[iM]:=m end
        else M[iM]:=M[iM+1]; iMmax:=iMmax−1;
        for j:=iM+1 step 1 until iMmax do M[j]:=M[j+1]
      end
    end
  end numbers below 100 in languages where units precede tenfolds;
  for iN:=1 step 1 until 15 do N[iN]:=0; pos:=0;
  for iM:=iMmax step−1 until 1 do
  begin m:=M[iM]; if m>31 then
    begin if m=41 then m:=(if lang=french then 39 else 42);
      if m>33 then pos:=m−30 else if m=32 then pos:=pos+
      2 else pos:=pos+(if (pos ÷ (if lang=chinese then 4
      else 3)) × (if lang=chinese then 4 else 3)=pos then
      3 else 1); if iM=1 ∨ m=32 then N[15−pos]:=1
    end power of ten gives new value to pos
```

```
      else if m<30 then
      begin if m<10 then N[15-pos]:=m else if m<20 then
         begin N[15-pos]:=m-(m÷ 10)×10; N[14-pos]:=1
         end numbers below 20 are put in N
         else N[14-pos]:=N[14-pos]+m-20
      end tenfolds are put in N
   end of the translation from back to front of M into N;
   for iN:=1 step 1 until 15 do
   begin if N[iN]≠0 then
      begin number of digits:=16-iN; goto Z end
   end determination of number of digits; goto SL; Z:
   for iN:=1 step 1 until 15 do
   begin if iN⩽15-number of digits then space else SYM(N[iN])
   end output of the decimal representation;
end read number name;

procedure fill W;
comment reads in the list of components. To each component is assigned: its
language (0=any language), its number of letters, and its meaning. The
limits for each language are stored in A;
begin integer h, H, i, iW, iWmax;
   iW:=1; H:=i:=0; iWmax:=108; A[5, 1]:=iWmax;
   lang:=0;
SW: h:=letter; if h=83 then
   begin A[lang, 1]:=iW-1; lang:=lang+1; A[lang, 0]:=iW;
      goto SW
   end;
   if h≠90 then
   begin i:=i+1; if i<5 then H:=H×26+h; goto SW end;
   W[iW, 1]:=H; W[iW, 2]:=lang; W[iW, 3]:=i; W[iW, 4]:=read;
   ABSFIXT(3, 0, W[iW, 4]); ABSFIXP(3, 0, W[iW, 4]);
   if iW≠iWmax then
   begin iW:=iW+1; H:=i:=0; goto SW end;
   A[0, 0]:=A[1, 0]; A[0, 1]:=iWmax
end fill W;
```

```
procedure write(s); string s;
comment This output procedure writes the string s;
begin PRINTTEXT(s); PUTEXT(s) end;

procedure write number name(N, lengthN, lang); array N;
integer lengthN, lang;
comment The number in array N with length lengthN is written in the language
lang;
begin integer i;
  if lang≠1 then NL;
  for i:=1 step 1 until (if lang=1 then 3 else 18) do space;
  if lang=chinese then CHINESE(N) else
  begin possible overlap:=overlap:=false;
    Next 3 digits ((lengthN−1) ÷ 3); if possible overlap then
    begin overlap:=true; NL; write('or with overlap:   ');
      Next 3 digits ((lengthN−1) ÷ 3);
    end overlapping case
  end non chinese language
end write number name;

procedure m(j, k); value j, k; integer j, k;
comment The word in row 9×j+k and the colum of the output language is
written;
begin switch morfemes:=m1, m2, m3, m4, m5, m6, m7, m8, m9, m11, m12,
  m13, m14, m15, m16, m17, m18, m19, m10, m20, m30, m40, m50, m60,
  m70, m80, m90, p2, p3, p6or4, p9or8, p12, m0;
  procedure P(du, ge, fr, en, ch); string du, ge, fr, en, ch;
  begin if lang=dutch then write(du) else if lang=german then
    write(ge) else if lang=french then write(fr) else if lang=
    english then write(en) else write(ch); goto WRITTEN
  end P;
  procedure Q(du, ge, fr, en); string du, ge, fr, en;
  P(du, ge, fr, en, ' ');
  procedure R(s); string s; P(s, s, s, s, s);
  goto morfemes[9×j+k];
```

```
comment Dutch       German       French          English       Chinese;
m 1:P ( 'een',      'ein',       'un',           'one',        'i');
m 2: if o1 then begin o1:=false; R(                            'liang') end else
     P ( 'twee',    'zwei',      'deux',         'two',        'erh');
m 3:P ( 'drie',     'drei',      'trois',        'three',      'san');
m 4:P ( 'vier',     'vier',      'quatre',       'four',       'ssu');
m 5:P ( 'vijf',     'fuenf',     'cinq',         'five',       'wu');
m 6:P ( 'zes',      'sechs',     'six',          'six',        'liu');
m 7:P ( 'zeven',    'sieben',    'sept',         'seven',      'ch'i');
m 8:P ( 'acht',     'acht',      'huit',         'eight',      'pa');
m 9:P ( 'negen',    'neun',      'neuf',         'nine',       'chiu');
m11:Q ( 'elf',      'elf',       'onze',         'eleven');
m12:Q ( 'twaalf',   'zwoelf',    'douze',        'twelve');
m13:Q ( 'dertien',  'dreizehn',  'treize',       'thirteen');
m14:Q ( 'veertien', 'vierzehn',  'quatorze',     'fourteen');
m15:Q ( 'vijftien', 'fuenfzehn', 'quinze',       'fifteen');
m16:Q ( 'zestien',  'sechzehn',  'seize',        'sixteen');
m17:Q ( 'zeventien', 'siebzehn', 'dix-sept',     'seventeen');
m18:Q ( 'achttien', 'achtzehn',  'dix-huit',     'eighteen');
m19:Q ( 'negentien', 'neunzehn', 'dix-neuf',     'nineteen');
m10:P ( 'tien',     'zehn',      'dix',          'ten',        'shih');
m20:Q ( 'twintig',  'zwanzig',   'vingt',        'twenty');
m30:Q ( 'dertig',   'dreissig',  'trente',       'thirty');
m40:Q ( 'veertig',  'vierzig',   'quarante',     'forty');
m50:Q ( 'vijftig',  'fuenfzig',  'cinquante',    'fifty');
m60:Q ( 'zestig',   'sechzig',   'soixante',     'sixty');
m70:Q ( 'zeventig', 'siebzig',   'soixante',     'seventy');
m80:Q ( 'tachtig',  'achtzig',   'quatre-vingt', 'eighty');
m90:Q ( 'negentig', 'neunzig',   'quatre-vingt', 'ninety');
p2  :P( 'honderd',  'hundert',   'cent',         'hundred',    'pai');
p3  :P( 'duizend',  'tausend',   'mille',        'thousand',   'ch'ien');
p6or4:P('miljoen',  'Million',   'million',      'million',    'wan');
p9or8: f:=¬f; if f ∧ lang=french then R('billion') else
     P( 'miljard',  'Milliarde', 'milliard',     'milliard',   'i4');
p12 :P( 'biljoen',  'Billion',   'trillion',     'billion',    'chao');
m0  :P( 'en',       'und',       'et',           ' ',          'ling');
```

```
WRITTEN:
end m(j, k);

procedure and; m(0, 33);
procedure space; write (' ');
procedure hyphen; write ('-');
procedure SYM(n); integer n;
comment If n is a digit, it is written by this output procedure;
begin PRSYM(n); PUSYM(n) end;
procedure NL; write ('
');

procedure Next 3 digits(i); integer i;
comment The i-th group from the back of three digits is produced, followed
by an appropriate power of thousand. It then calls itself with lowered i until
i becomes 1;
begin integer i3;
  i3: = i × 3; if ¬ overlap then
  begin if ¬ possible overlap then possible overlap: = i≠0 ∧ N[13−
    i3] = 0 ∧ N[14−i3] = 0 ∧ N[15−i3] ≠ 0 ∧ (lang = french →
    N[15−i3] = 1) ∧ N[16−i3] ≠ 0;
  NON OVERLAP: hundredfold(0, N[13−i3]);
    if N[13−i3] ≠ 0 then
    begin if lang = french then
      begin if N[14−i3] + N[15−i3] = 0 ∧ N[13−i3] ≠ 1 then
        write ('s'); space
      end cents
    end first of the 3 digits;
    from 1 up to 100(N[14−i3], N[15−i3]);
    if i = 0 ∧ lang = german ∧ N[14] = 0 ∧ N[15] = 1 then write ('s');
    one: = (N[13−i3] = 0 ∧ N[14−i3] = 0 ∧ N[15−i3] = 1);
    if i = 0 then goto MADE;
    if N[13−i3] + N[14−i3] + N[15−i3] ≠ 0 then
    thousand to the power(i); Next 3 digits(i−1)
  end of non overlapping case
```

```
else
begin if i≠0 ∧ N[13−i3]=0 ∧ N[14−i3]=0 ∧ N[15−i3]≠0
  ∧ (lang=french → N[15−i3]=1) ∧ N[16−i3]≠0 then
  begin hundredfold (N[15−i3], N[16−i3]);
    if lang=french then
    begin if N[17−i3]+N[18−i3]=0 then write ('s'); space
    end cents;
    from 1 up to 100 (N[17−i3], N[18−i3]);
    if i=1 ∧ N[14]=0 ∧ N[15]=1 ∧ lang=german then write (
    's'); one:=false; if i=1 then goto MADE;
    thousand to the power (i−1); Next 3 digits (i−2)
  end overlapping case
  else goto NON OVERLAP
end;
MADE:
end Next 3 digits (i);

procedure from 1 up to 100 (j, k); value j, k; integer j, k;
comment produces 10 × j+k;
if k≠0 then
begin if j<2 then m(j, k) else
  begin if lang=french ∨ lang=english then
    begin m(2, j); if lang=english then
      begin hyphen; m(0, k) end
      else
      begin if j≠7 ∧ j≠9 then
        begin if k=1 then
          begin if j≠8 then and end
          else hyphen; m(0, k)
        end
        else
        begin if k=1 ∧ j=7 then and else hyphen;
          from 1 up to 100 (1, k)
        end
```

```
        end French up to 100
      end French and English
      else
      begin m(0, k); and; m(2, j) end Dutch and German
    end above 20
end
else
begin if j≠0 then
    begin m(2, j); if lang=french ∧ j>6 then
      begin if j=8 then write('s') else
        begin hyphen; from 1 up to 100(1, 0) end
      end French 70, 80, 90
    end tenfolds
end 1 up to 100;

procedure hundredfold(j, k); value j, k; integer j, k;
comment produces (10×j+k)×100;
begin if j+k≠1 ∨ lang=english then from 1 up to 100(j, k);
    if k≠0 then m(3, 1)
end hundredfold;

procedure thousand to the power(k); value k; integer k;
comment produces 1000 ↑ k;
begin if k>1 then
    begin if one ∧ lang=german then write('e') end;
    m(3, k+1); if k>1 then
    begin if ¬ one then
      begin if lang=french then write('s')
        else if lang=german then
        begin if k≠3 then write('e'); write('n') end
      end;
      if lang≠dutch then space
    end
end 1000 to the power(k);
```

```
procedure CHINESE(Number); array Number;
comment The Chinese name for the number in array Number is produced;
begin integer i, shih, pai, chien, wan, i4, chao, j, L, LL;
  Boolean change;
  integer array S[-1:36];
  comment The string of Chinese morfemes with length L is stored in S,
  numbers below 10 in their natural code, power of ten as 10, 11, 12, 13, 14, 15
  (chao);

  procedure write chinese;
  comment The string of Chinese morfemes in S is produced;
  for j:=1 step 1 until L do
  begin if S[j]=0 then and else if S[j]<10 then m(0, S[j])
    else if S[j]=10 then m(0, 19) else m(2, S[j]-1); space
  end;

  comment The 7 obligatory transformations:   ;

  procedure D(condition for context of ling, deletend);
  Boolean condition for context of ling; integer deletend;
  for j:=1 step 1 until L do
  begin if ling(j) then
    begin if condition for context of ling then delete(deletend) end
  end general deletion transformation around ling;

  procedure delete(n); value n; integer n;
  begin integer k;
    L:=L-1;
    for k:=n step 1 until L do S[k]:=S[k+1]; change:=true
  end delete;

  procedure D1; D(shih pai chien(j-1) ∧ wan i4 chao(j+1), j);

  procedure D2;
  D((shih pai chien(j-1) ∨ wan i4 chao(j-1))
   ∧ shih pai chien(j+1), j+1);
```

```
procedure D3; D(ling(j+1), j+1);

procedure D4; D(chao i4(j−1) ∧ i4 wan(j+1), j+1);

procedure D5;
if ling(1) ∧ L>2 ∧ (shih pai chien(2) ∨ wan i4 chao(2)) then
begin delete(1); delete(1) end;

procedure D6; if ling(L) ∧ L>1 then delete(L);

procedure D7; if S[1]=i ∧ S[2]=shih then delete(1);

comment The 5 optional transformations:  ;

procedure option(n); integer n;
comment if the application of an optional transformation is possible the
alternative form is produced;
begin NL; write('or with option o'); SYM(n); write(':');
  write chinese
end option;

procedure O1;
if S[1]=2 ∧ (wan i4 chao(2) ∨ (shih pai chien(2) ∧ S[2]≠shih))
then
begin o1:=true; option(1) end;

procedure O2;
begin integer a, b, c;
  a:=S[L−2]; b:=S[L−1]; c:=S[L];
  if ((a=chien ∧ c=pai) ∨ (a=pai ∧ c=shih) ∨ (a=wan ∧
  c=chien)) ∧ b>0 ∧ b<10 then
  begin delete(L); option(2) end
end O2;
```

```
procedure O3;
for j:=2 step 1 until L-1 do
begin if S[j]=shih then
  begin if S[j-1]=i ∧ S[j+1]>0 ∧ S[j+1]<10 then
    begin delete(j-1); option(3); O3 end
  end
end;

procedure O4;
begin change:=false;
  D(S[j+1]<10 ∧ S[j+2]=pai ∧ ling(j+3) ∧ S[j+4]<
  10,j); if change then option(4)
end O4;

procedure O5;
begin integer jj;
  change:=false;
  for j:=2 step 1 until L-4 do
  begin if chaoi4(j) then
    begin if ling(j+1) then
      begin for jj:=2 step 1 until j-1 do
        begin if chaoi4(jj) ∨ ling(jj) then goto OUT end;
        for jj:=L step-1 until j+3 do
        begin if i4wan(jj) then goto OUT; if ling(jj) then
          begin delete(j+1); goto OUT end
        end investigation of string to the right of ling
      end chao i4 followed by ling
    end occurrence of chao i4;
  OUT:
  end investigation of string;
  if change then option(5)
end O5;
```

```
    Boolean procedure ling(n); integer n; ling:=S[n]=0;
    Boolean procedure shih pai chien(n); integer n;
    shih pai chien:=S[n]≥shih ∧ S[n]≤chien;
    Boolean procedure wan i4 chao(n); integer n;
    wan i4 chao:=S[n]≥wan;
    Boolean procedure chao i4(n); integer n;
    chao i4:=S[n]≥i4;
    Boolean procedure i4 wan(n); integer n;
    i4 wan:=S[n]=i4 ∨ S[n]=wan;

    i:=1; shih:=10; pai:=11; chien:=12; wan:=13; i4:=14; chao:=15;
    for j:=-1, 0, 32, 33, 34, 35, 36 do S[j]:=-1; S[1]:=0;
    for j:=3 step 2 until 31 do S[j]:=Number[j÷2];
    for j:=2 step 8 until 26 do
    begin S[j]:=chien; S[j+2]:=pai; S[j+4]:=shih end;
    S[8]:=chao; S[16]:=i4; S[24]:=wan; L:=31;
  leading lings: change:=false; D5;
    if change then goto leading lings;
  TRANSFORM: change:=false; LL:=L; D(true, 32); if change then
    begin L:=LL; change:=false; D1; D2; D3; D4; D6;
      if change then goto TRANSFORM
    end obligational transformations in connection with ling;
    D7; write chinese; O1; O2; O3; O4; O5;
  end chinese number

  With the above procedures translation programs of many kinds can be
  written. One of them, reading in any language and translating into all five
  languages, is:;
  o1:=f:=false; fillW; lang:=0;
  dutch:=1; german:=2; french:=3; english:=4; chinese:=5;
START: NL; read number name;
  for lang:=dutch, german, french, english, chinese do
  write number name(N, number of digits, lang); goto START
end
```

The input text is repeated, in italics, in the following output:

acht 8 *ard* 52 *billi* 54 *e* 0 *elf* 11 *en* 30 *hundred* 32 *milli* 53 *on* 51 *s* 0 *six* 6 *vier* 4

bilj 54 *der* 3 *drie* 3 *duizend* 33 *een* 1 *honderd* 32 *milj* 53 *negen* 9 *oen* 51 *tachtig* 28 *tien* 10 *tig* 20 *twaalf* 12 *twee* 2 *twintig* 22 *veer* 4 *vijf* 5 *zes* 6 *zeven* 7

drei 3 *ein* 1 *fuenf* 5 *funf* 5 *neun* 9 *sech* 6 *sieb* 7 *ssig* 20 *tausend* 33 *und* 30 *zehn* 10 *zig* 20 *zwanzig* 22 *zwei* 2 *zwolf* 12 *zwoelf* 12

cent 32 *cinq* 5 *deux* 2 *dix* 10 *douze* 12 *et* 0 *huit* 8 *mille* 33 *neuf* 9 *onze* 11 *seize* 16 *sept* 7 *soixante* 26 *treize* 13 *trente* 23 *trillion* 42 *trois* 3 *quante* 20 *quatre* 4 *quatorze* 14 *quarante* 24 *quinze* 15 *un* 1 *vingt* 22

and 30 *eigh* 8 *eleven* 11 *fif* 5 *five* 5 *for* 4 *four* 4 *nine* 9 *one* 1 *seven* 7 *t* 0 *teen* 10 *ten* 21 *thir* 3 *thousand* 33 *three* 3 *twelve* 12 *twenty* 22 *two* 2 *ty* 20

chao 42 *ch'i* 7 *ch'ien* 33 *chiu* 9 *erh* 2 *i* 1 *i4* 38 *liang* 2 *ling* 0 *liu* 6 *pa* 8 *pai* 32 *san* 3 *shih* 10 *ssu* 4 *wan* 34 *wu* 5

eenmiljard tweehonderd miljoen zeshonderdduizend vijfhonderd

	1200600500	eenmiljardtweehonderdmiljoenzeshonderdduizend vijfhonderd
or with overlap:		twaalfhonderdmiljoenzeshonderdduizendvijfhonderd
		eine Milliarde zweihundert Millionen sechshunderttausendfuenfhundert
or with overlap:		zwoelfhundert Millionen sechshunderttausendfuenf hundert
		un billion deux cents millions six cents mille cinq cents
or with overlap:		douze cents millions six cents mille cinq cents
		one milliard two hundred million six hundred thousand five hundred
or with overlap:		twelve hundred million six hundred thousand five hundred
		shih erh i4 ling liu shih wan ling wu pai
or with option o5:		shih erh i4 liu shih wan ling wu pai

pa i4 ling i pai i shih pa

	800000118	achthonderdmiljoenhonderdachttien
		achthundert Millionen hundertachtzehn
		huit cents millions cent dix-huit

	eight hundred million one hundred eighteen
	pa i4 ling i pai i shih pa
or with option o3:	pa i4 ling i pai shih pa
seven billion	
7000000000000	zevenbiljoen
	sieben Billionen
	sept trillions
	seven billion
	ch'i chao
sept billion	
7000000000	zevenmiljard
	sieben Milliarden
	sept billions
	seven milliard
	ch'i shih i4
siebenhundertsiebenundsiebzig	
777	zevenhonderdzevenenzeventig
	siebenhundertsiebenundsiebzig
	sept cent soixante-dix-sept
	seven hundred seventy-seven
	ch'i pai ch'i shih ch'i
pa pai chiu	
890	achthonderdnegentig
	achthundertneunzig
	huit cent quatre-vingt-dix
	eight hundred ninety
	pa pai chiu shih
or with option o2:	pa pai chiu
five-and-twenty	
25	vijfentwintig
	fuenfundzwanzig
	vingt-cinq
	twenty-five
	erh shih wu
?	

This example was produced on the Electrologica X8 of the Mathematical Centre in 5 seconds.

9175