STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

MR 68

Automatic Translation of Numbers Into Dutch

H. Brandt Corstius

1965

FOUNDATIONS OF LANGUAGE

INTERNATIONAL JOURNAL OF LANGUAGE AND PHILOSOPHY

H. BRANDT CORSTIUS

AUTOMATIC TRANSLATION OF NUMBERS INTO DUTCH*

The algorithmical language ALGOL 60[1] is, contrary to what is sometimes believed, very suitable for linguistic uses. A generative grammar of the Chomskian type, e.g.

sentence→A+B+C
A→d+e
B→A+f
C→B+A

may be written without any further knowledge of computer programming as an ALGOL-program:

```
begin procedure sentence;
        begin A; B; C end;
    procedure A;
        begin outstring (d); outstring (e) end;
    procedure B;
        begin A; outstring (f) end;
    procedure C;
        begin B; A end;
    sentence
end
```

More complicated grammars with choices in the right hand sides of the replacement rules, indices, and transformational rules, can be similarly programmed.

As an example we give here a program, writing the names of natural numbers in the Dutch language, inspired by a grammar of Van Katwijk.[2] The program takes natural numbers up to 10^{66}, which number might be called by extrapolation "undeciljoen". The program was run successfully on the Electrologica X1 computer of the Mathematical Centre in Amsterdam using an ALGOL translator constructed by E. W. Dijkstra and J. A. Zonneveld.

* Communication MR68 of Mathematical Centre, Amsterdam.

1 Naur, P. *et al.*, 'Revised Report on the algorithmic language ALGOL 60', *Numerische Mathematik* **4** (1963) 420–453; *Communications of the Association for Computing Machinery* **6** (1963) 1–17; Regnecentralen, Copenhagen 1962.

2 A. F. V. van Katwijk, 'A Grammar of Dutch Number Names', this issue, pp. 51–58.

H. BRANDT CORSTIUS

ALGOL PROGRAM FOR TRANSLATION OF DIGITAL NUMBERS INTO DUTCH

```
begin comment Natural numbers given in digits are written in Dutch. The manner of input
      of the digits and output of the words is irrelevant. Therefore the two procedures Take
      Next Number and Write are separated from the linguistic part of the program. Their
      effect is described in their respective comments, their content may be omitted.
      The undeclared procedure stop stops the machine, the undeclared procedure NLCR
      gives New Line Carriage Return;

integer i, Number Of Digits;
Boolean overlap; comment Some numbers can be written in two ways (1200 = twaalf-
        honderd or eenduizendtweehonderd);
integer array Number [1:66];

procedure Take Next Number;
    comment This inputprocedure puts the digits of the given number in the back of
    the array Number, with zeroes in front of it. The value of Number Of Digits is
    determined. If the number of digits is greater than 66 this fact is communicated
    and the oversized number is omitted;
    begin integer digit;
        integer procedure Next Symbol;
            comment If the next symbol is a digit Next Symbol gets the value of
            this digit. If the next symbol is a number separator Next Symbol be-
            comes 10. If the next symbol indicates the end of the list of numbers
            (end) the program stops;
        begin integer k;
            insymbol (1, '1234567890end', k); if k = 11 then stop; Next Symbol: =
            if k < 0 then Next Symbol else if k = 0 then 10 else if k = 10 then 0
            else k end;
        i:= 1;
READ NUMBER: digit:= Next Symbol;
        if digit = 0 ∧ i = 1 then goto READ NUMBER;
        if digit = 10 then goto NUMBER READ;
        if i = 67 then begin Write ('number of digits greater than 66');
OMIT:   if Next Symbol = 10 then goto START else goto OMIT end;
        Number [i]:= digit; i:= i + 1; goto READ NUMBER;
NUMBER READ: if i = 1 then goto READ NUMBER;
        Number Of Digits:= i − 1;
        for i:= 66 step − 1 until 1 do Number [i]:= if i > 66 − Number Of
        Digits then Number [i − 66 + Number Of Digits] else 0;
        end Take Next Number;

    procedure Write (text); string text;
        comment This outputprocedure writes the string text;
        outstring (2, text);

    procedure Next 3 digits (i);
        integer i;
        begin integer i3;
            procedure choose (j, k);
            comment one of the 37 words is chosen;
            integer j, k;
            begin switch words:= w 1, w 2, w 3, w 4, w 5, w 6, w 7, w 8, w 9, w 10,
            w 11, w 12, w 13, w 14, w 15, w 16, w 17, w 18, w 19, w 20, w 21, w 22,
            w 23, w 24, w 25, w 26, w 27, w 28, w 29, w 30, w 31, w 32, w 33, w 34,
            w 35, w 36, w 37;
```

```
        procedure P (word);
            string word;
            begin Write (word); goto WRITTEN end;
        goto words [9 × j + k];
w 1  : P ('een');          w 2  : P ('twee');        w 3  : P ('drie');
w 4  : P ('vier');         w 5  : P ('vijf');        w 6  : P ('zes');
w 7  : P ('zeven');        w 8  : P ('acht');        w 9  : P ('negen');
w 10: P ('elf');           w 11: P ('twaalf');       w 12: P ('dertien');
w 13: P ('veertien');      w 14: P ('vijftien');     w 15: P ('zestien');
w 16: P ('zeventien');     w 17: P ('achttien');     w 18: P ('negentien');
w 19: P ('tien');          w 20: P ('twintig');      w 21: P ('dertig');
w 22: P ('veertig');       w 23: P ('vijftig');      w 24: P ('zestig');
w 25: P ('zeventig');      w 26: P ('tachtig');      w 27: P ('negentig');
w 28: P ('milj');          w 29: P ('bilj');         w 30: P ('trilj');
w 31: P ('kwadrilj');      w 32: P ('kwintilj');     w 33: P ('sextilj');
w 34: P ('septilj');       w 35: P ('octilj');       w 36: P ('nonilj');
w 37: P ('decilj');                                  WRITTEN:
    end choose;
procedure from 1 up to 100 (j, k); comment writes 10 × j + k;
    integer j, k;
    if k ≠ 0 then begin if j < 2 then choose (j, k) else
            begin choose (0, k); Write ('en'); choose (2, j) end end
        else if j ≠ 0 then choose (2, j);
procedure hundredfold (j, k); comment writes (10 × j + k) × 100;
    integer j, k;
    begin if j + k ≠ 1 then from 1 up to 100 (j, k);
    if k ≠ 0 then Write ('honderd') end hundredfold;
        procedure thousand to the power (k); comment writes (1000) ↑ k;
            integer k;
            if k = 0 then goto START else
            begin if k = 1 then Write ('duizend') else
                begin choose (3, k ÷ 2); if k ÷ 2 × 2 = k then
                Write ('oen') else Write ('ard') end end;
        i3 := 3 × i;
        if overlap ∧ Number [66 − i3] × (if i = 0 then 0 else Number [67–i3]) ≠ 0
        ∧ Number [65–i3] + Number [64–i3] = 0 then
        begin hundredfold (Number [66–i3], Number [67–i3]);
            from 1 up to 100 (Number [68–i3], Number [69–i3]);
            thousand to the power (i − 1);
            Next 3 digits (i − 2)
        end overlapping case else
        begin hundredfold (0, Number [64–i3]);
            from 1 up to 100 (Number [65–i3], Number [66–i3]);
            if Number [64–i3] + Number [65–i3] + Number [66–i3] ≠ 0 ∨ i = 0
            then thousand to the power (i);
            Next 3 digits (i − 1)
        end non-overlapping case
        end Next 3 digits;
Write ('Result of program writing Dutch number names');
    overlap := true;
START: overlap := ¬ overlap; NLCR;
    Take Next Number;
    i := (Number Of Digits − 1) ÷ 3;
    Next 3 digits (i)
end
```

The reverse process, from words to digits, is easily programmable. If, in addition, we have translation programs for number names in other languages, mechanical translation of number names from one natural language into another is possible, with the digital representation as an intermediate language. (We do not suggest that it is possible or desirable to find such an intermediate language in Mechanical Translation in general.)

The program should be readable without further explanation. It consists essentially of one statement: the call of the procedure "Next 3 digits (i)" which calls itself with a lowered value of the parameter i.

This program may show that an unambiguous language for exact description of complicated processes, as ALGOL 60 or any other general programming language, has its value also outside Numerical Mathematics.

Mathematisch Centrum, Amsterdam

FOUNDATIONS OF LANGUAGE

INTERNATIONAL JOURNAL OF LANGUAGE AND PHILOSOPHY

FOUNDATIONS OF LANGUAGE invites general or technical contributions, original or expository, in fields of research dealing with language and the foundations of its study. Normally only articles in English will be accepted.

Articles for publication and communications relative to editorial matters should be sent to JOHN W. M. VERHAAR, Managing Editor, Le Moyne College, Le Moyne Heights, Syracuse, N.Y. 13214, U.S.A. After June 1, 1965 his address will be: Ateneo de Manila, P.O. Box 154, Manila, Philippine Islands.

Book reviews and books submitted for review should be sent to J. F. STAAL, Review editor, Instituut voor Filosofie, Universiteit van Amsterdam, Amsterdam, The Netherlands.

All manuscripts should be typewritten with wide margin and double spacing between the lines, and the author should keep a complete copy; footnotes should be numbered consecutively and should also be typed with wide margin and double spacing (preferably on a separate sheet). Authors receive fifty reprints of their article or book-review. Additional reprints are obtainable from the publisher.

FOUNDATIONS OF LANGUAGE is published quarterly. The subscription price is *f* 42.— ($ 11.75; 84s.) per volume of four issues. Subscriptions, changes of address and all business correspondence should be addressed to D. REIDEL PUBLISHING COMPANY, P.O. Box 17, Dordrecht, The Netherlands.

Printed in The Netherlands by D. Reidel, Dordrecht

CONTENTS