An algol program for deciding derivability in minimal propositional calculus with implication and conjunction over a three letter alphabet

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics

Memorandum 1975-06

Issued May, 1975

AN ALGOL PROGRAM FOR DECIDING DERIVABILITY IN MINIMAL PROPOSITIONAL CALCULUS WITH IMPLICATION AND CONJUNCTION OVER A THREE LETTER ALPHABET

by

N.G. de Bruijn

University of Technology
Department of Mathematics
PO Box 513, Eindhoven
The Netherlands

An Algol program for deciding derivability in minimal propositional calculus with implication and conjunction over a three letter alphabet

by N.G. de Bruijn.

The questions of derivability in the logical system mentioned in the title can be answered completely by means of a partially ordered set of 61 points (see [1]). There are 623 662 965 552 330 equivalence classes (two logical expressions are called equivalent if they can be derived from each other). These classes correspond one-to-one to the 623 662 965 552 330 closed sets in the 61-point partially ordered set. Every closed set can be completely described by its minima, which form an independence subset of the 61-point set. The one-to-one mapping can be described by means of the mapping ℓcv: if α is an expression then $\ell cv(\alpha)$ is an independence subset. It is called "lower carrier" of α. We have $\ell cv(\alpha) = \ell cv(\beta)$ if and only if α and β are logically equivalent. The closure of $\ell cv(\alpha)$ is denoted by $v(\alpha)$; $v(\alpha)$ is a valuation, mapping expressions to closed sets according to the rules

$$v(\alpha \wedge \beta) = v(\alpha) \cup v(\beta),$$
$$v(\alpha \rightarrow \beta) = c\ell(v(\beta) \setminus v(\alpha))$$

(where $c\ell$ stands for "closure").

The partially ordered set was displayed on page 21 of [1]; it is reproduced on page 5 of the present Memorandum.

The atomic expressions a, b, c correspond to closed sets, described by

$\ell cv(a) = \{2,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,22,23,24,41,43,45,61\}$

$\ell cv(b) = \{1,3,5,22,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,42,43,44,61\}$

$\ell cv(c) = \{2,3,4,21,23,25,42,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61\}$

The computer program we presently describe accepts logical expressions in a coded form, and determines their ℓcv's.

The program handles 251 expressions, numbered 0,...,250, but the number 250 can of course be replaced by any other number (provided that memory space is available). Expression 0 is used as working space, expressions 1, 2, 3 are reserved for the atoms a, b, c.

The input data have to be a sequence of integers m, p_1, q_1, r_1, k_1, p_2, q_2, r_2, k_2,...,p_n, q_n, r_n, k_n. The program has the effect that, for $i = 1,...,n$ the following operation is carried out on the numbers p_i, q_i, r_i, k_i: the expression numbered r_i is formed as $\alpha \to \beta$ if $k_i = 0$, and as $\alpha \wedge \beta$ otherwise. Here α, β are the expressions bearing numbers p_i, q_i. It is required that $1 \le p_i \le 250$, $1 \le q_i \le 250$, $1 \le r_i \le 250$, and that $r_i \neq p_i$, $r_i \neq q_i$. The program produces an output consisting of

(i) a record of what it has been considering:

$$p_i \to q_i = r_i \quad \text{or} \quad p_i \wedge q_i = r_i$$

(ii) the elements of the ℓcv of r_i.

E.g., if the input is

5,1,3,4,1, 4,2,5,0, 1,2,6,0, 6,1,7,0, 1,3,8,0

then the output becomes(apart from the content of ℓcv(a), ℓcv(b), ℓcv(c), which always preceeds all further output)

```
1 ∧ 3 = 4    expr 4 has lower carrier  2  3  4  21  23  41  43  45  61
4 → 2 = 5    expr 5 has lower carrier  1
1 → 2 = 6    expr 6 has lower carrier  1  3  5
6 → 1 = 7    expr 7 has lower carrier  22 23 24 41  43  61
1 → 3 = 8    expr 8 has lower carrier  3  4  21
```

Note that this refers to the following expressions:

$\alpha_4 = a \wedge c$

$\alpha_5 = (a \wedge c) \rightarrow b$

$\alpha_6 = a \rightarrow b$

$\alpha_7 = (a \rightarrow b) \rightarrow a$

$\alpha_8 = a \rightarrow c.$

We can go on, building any expression we like. Whenever we present a derivable expression the computer shows this by giving an empty lower carrier.

The program was written for a particular computer system (the MC system for the Philips-Electrologica X 8) with the following input-output procedures:

(i) The integer procedure "read" for reading an integer from the output data.

(ii) printtext (‡ ‡) for printing the string between ‡ ‡.

(iii) absfixt (k,0,h) for printing the value of the integer represented by the identifyer h (to k decimal places).

(iv) nℓcr for "new line, carriage return".

(v) space (k) for leaving k open spaces.

We add a few words on the interpretation of the arrays over[1:61,1:61] and word[0:250,1:61]. If $1 \leq i \leq 61$, $1 \leq j \leq 61$, then over[i,j] is to have the value 1 if $j \leq i$ in the partial order of the 61-point set, and the value 0 otherwise.

The array elements word[i,1],...,word[i,61] represent the lower carrier of the i-th expression. We have word[i,j] = 1 if point number j belongs to the lower carrier of expression number i, and word[i,j] = 0 otherwise.

The program takes care that the array "over" gets the values required for the 61-point partially ordered set, and it sees to it that expressions numbered 1, 2, 3 get the proper lower carriers.

The time needed for the processing of a quartuple p_i, q_i, r_i, k_i on the X 8 computer is of the order of 10^{-1} sec.

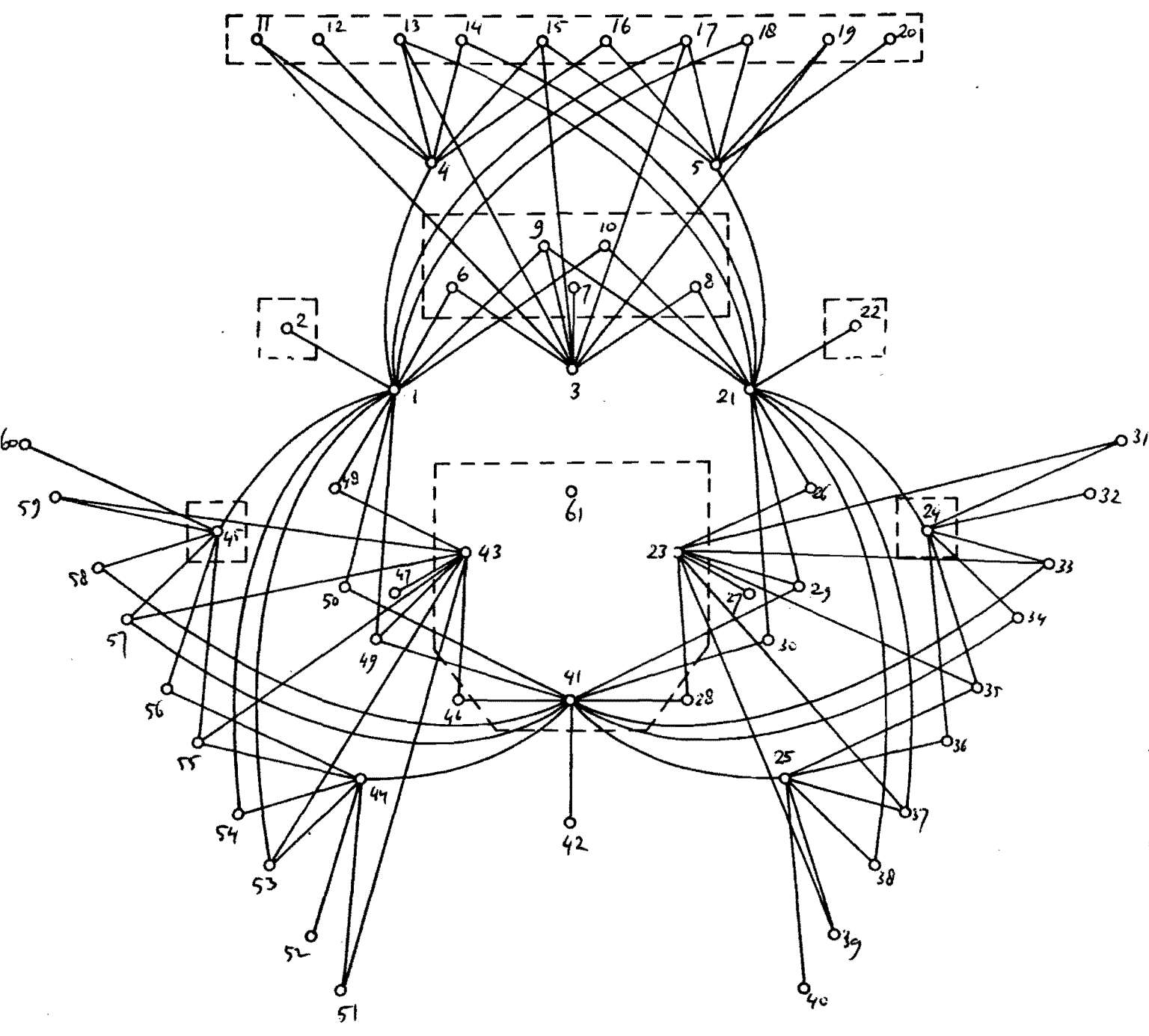
The complete Algol program is presented on pages 6 and 7 of this note.

If we do not bother about time consumption, the key procedure "implic" can be given a much simpler body, e.g.:

```
begin integer i,j; for j:=1 step 1 until 61 do
        begin word[z,j]:= word[y,j];
              for i:=1 step 1 until 61 do
              if word[x,i]=1∧ over[j,i]=1 then word[z,j]:=0
        end
end;
```

Reference:

1 N.G. de Bruijn, Exact finite models for minimal propositional calculus over a finite alphabet, Technological University Eindhoven, Department of Mathematics, T.H.Report 75-WSK-02, February 1975.



This graph represents the 61-point partially ordered set. If $x \leq y$ then we can proceed from x to y using lines of the graph in outward direction (away from the centre 61) only.

The points of $\ell cv(a)$ are surrounded by interrupted lines. Those of $\ell cv(b)$ and $\ell cv(c)$ are obtained by rotation over 120° and 240°.

```
    begin integer array over [1:61,1:61], word[0:250,1:61]; integer i,j,k,p,q,r,nr;

    procedure printword(h); value h; integer h;
    begin integer i; printtext(‡expr‡); absfixt(3,0,h); printtext(‡has lower carrie
        for i:=1 step 1 until 61 do if word[h,i]=1 then absfixt(2,0,i)
    end;

    procedure implic(x,y,z); value x,y,z; integer x,y,z;
    begin integer i,j,s; for j:=1 step 1 until 61 do
    begin s:=0; if word[y,j]=1 then
        begin if word[x,j]=1 then goto end;
          if j<21 then
                begin for i:=1,21,3,4,5 do if word[x,i]=0 then
                      else if over [j,i]=1 then goto end
                end
          else if j<41 then
                begin for i:=21,41,23,24,25 do if word[x,i]=0 then
                      else if over[j,i]=1 then goto end
                end
          else for i:=41,1,44,43,45 do if word[x,i]=0 then
          else if over[j,i]=1 then goto end; s:=1;
end:    end;

    word[z,j]:=s
    end
    end;

    procedure conju(x,y,z); value x,y,z; integer x,y,z;
    begin implic(x,y,z); implic(z,x,0); dirsum(0,z) end;
```

```
procedure dirsum(x,y); value x,y; integer x,y;
begin integer i;
     begin for i:=1 step 1 until 61 do
          if word[x,i]=1 then word[y,i]:=1
     end
end;

integer procedure plus(i,j); value i,j; integer i,j;
plus := if i+j>60 then i+j-60 else i+j;

procedure fillover(p,q); value p,q; integer p,q;
begin  over[p,q]:= over[plus(p,20),plus(q,20)]:= over[plus(p,40),plus(q,40)]:=1
end;

for p:=1 step 1 until 61 do for q:=1 step 1 until 61 do
                        over[p,q]:= if p=q then 1 else 0;
for i:=2,4,6,9 step 1 until 18,45,48,49,50,53 step 1 until 60 do fillover(i,1);
for i:=6,7,8,9,11,13,15,17,19 do fillover(i,3);
for i:=11,12,13,14,15,16 do fillover(i,4);
for i:=15,16,17,18,19,20 do fillover(i,5);

for i:=1 step 1 until 61 do word[1,i]:=word[2,i]:=word[3,i]:= if i=61 then 1
else 0;
for i:=2,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,22,23,24,41,43,45 do
word[1,i]:=word[2,plus(i,20)]:=word[3,plus(i,40)]:=1;
nlcr;printword(1);nlcr;printword(2);nlcr;printword(3);

nr:=read; for i:=1 step 1 until nr do
      begin p:=read; q:=read; r:=read; k:=read; nlcr;nlcr;
           absfixt(4,0,p); if k=0 then printtext(‹->›) else printtext(‹∧›);
           absfixt(3,0,q); printtext(‹=›);absfixt(4,0,r); space(4);
           if k=0 then implic(p,q,r) else conju(p,q,r); printword(r)
      end

end
```