ソフトウェア工学 22－14
（1982. 2. 10）

自動合成系LPSのあるプログラミング環境

大村伸一（京都産大・理）
謝　章文（京都産大・計科研）

1. はじめに

論理的基礎を持ったプログラムの自動合成系として我々の開発してきたLPS (Logical Program Synthesis) pilot systemsは，これまでの実験を通して自動合成系が十分に強力であり，効率上も満足のゆくものであり得ることを示した。[2～6]

自動合成系が現在の手工業的プログラミングにとって代わった場合，そのプログラミング環境におけるプログラムの開発工程は図1のような形態となるであろう。この環境での人間活動は「目的・システム要求」の把握，「仕様書の作成」，「合成されたプログラムの実行評価」に限定される。特に，現在プログラムが占めている中心的な役割は仕様書へと移行し，作成されるシステム（プログラム）は，この仕様書の内容だけによって影響を受ける。一方，プログラム自体は特に必要がなければ触れる必要はなく，この環境で働く人間にとって，特定の「プログラム言語」（「手続き向き言語」）の重要性が薄らぐ。

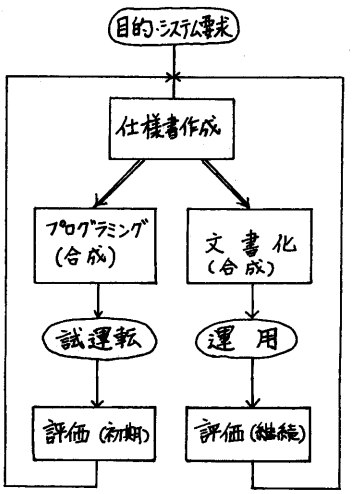

図1　合成系のあるプログラミング環境でのプログラム開発工程

1977年以来，我々はLPS pilot systemsの開発を続けてきたが，現在その環境は，図1の文書化に関する部分の自動化を除き，合成系のあるプログラミング環境の1つのミニチュアになっている。この論文では，1979年に開発されたpilot system F1を用いて，この環境でのプログラム開発状況の一例を示す。

2. 新環境におけるプログラム開発例

2.1　例題の選定

新環境においては，合成されるプログラムの仕様書に対する正当性が保証されている。このため，仕様書が目的・システム要求に対して正しいかどうかが極めて重大になってくる。

この環境では，目的・システム要求に対する仕様書の不適切さすなわち論理的エラーを取り除く事が，プログラム作成の最も中心的活動の1つとなる。しかし，問題が簡単な場合には論理エラーの入り込む余地はほとんどない。当初我々は，数理パズルのプログラム（仕様書40行，プログラム250行(FORTRAN)）を作成する上での論理エラーを修正する過程を提示する予定であったが，紙数の制限からより小さな例題を選ばざるを得なくなった。そのため，ここでは論理エラーを扱う例の

かわりに，それとよく類似した，あるプログラムを異なる解法や構造へ等価変換する過程を取りあげる。問題としては"LONGEST UPSEQUENCE"を取り上げる。これは1979年，BRUSSELSで行われたIFIP WG2.1 Meetingでも例題として取り上げられている，よく知られた問題である。

```
Example 7A --- LONGEST UPSEQUENCES

Given a sequence of n integers, A[0],A[1]..A[N-1], an upsequence is a
subsequence which is orderd in ascending order.  A subsequence is any
subset of the orisinal sequence where the orisinal order is retained.
(There are 2**N possible subsequences). Ordered in ascendins order means
that no element of the upsequence has a right hand neishbour smaller than
itself.

Give an program which, siven a sequence, computes the lensth of its longest
upsequence.

Note that all subsequences of lensth 1 are upsequences by this definition.

There may be more than one longest upsequence havins the same lensth, for
example the sequence (3,1,1,2,5,3) yields 4 for the maxinum lensth, realized
either by (1,1,2,5) or (1,1,2,3).
```

2.2 LPS仕様とその合成プログラム

まず，binary searchを行うclosedサブルーチンを持つようなプログラムの作成を考える。これはDijkstra[1]に問題に対する詳しい考察があるのでそれを参照されたい。また，ここでは仕様の細い理解が目的ではないので，詳細は説明しない。

```
<<  Specification  >>

~Conjecture :

-1. (NN Z) -L(NN,Z)

Axioms :

2. (Z2 H2) -P(NN,Z2,H2,!M1(H2)) v L(NN,Z2)
3. () P(1,1,1,A(0))
4. (N4 K4 H4) -<=(!M1(K4),A(N4)) v -P(N4,K4,H4,!M1(H4)) v =(N4,NN) v P(ADD(N4,1)
,ADD(K4,1),ADD(K4,1),A(N4))
5. (N5 K5 H5) -<(A(N5),!M1(1)) v -P(N5,K5,H5,!M1(H5)) v =(N5,NN) v P(ADD(N5,1),
K5,1,A(N5))
6. (N6 K6 J6 H6) -<=(!M1(1),A(N6)) v -P(N6,K6,H6,!M1(H6)) v -Q(N6,1,K6,J6) v =(
N6,NN) v -<(A(N6),!M1(K6)) v P(ADD(N6,1),K6,J6,A(N6))
7. (N7 H7 I7 J7 K7) R(N7,H7,K7,I7,DIV(ADD(I7,ADD(J7,1)),2)) v -R(N7,H7,K7,I7,J7)
 v ->(!M1(DIV(ADD(I7,ADD(J7,1)),2)),A(N7))
8. (N8 H8 I8 J8 K8) R(N8,H8,K8,DIV(ADD(I8,ADD(J8,1)),2),J8) v -R(N8,H8,K8,I8,J8)
 v -<=(!M1(DIV(ADD(I8,ADD(J8,1)),2)),A(N8))
9. (N9 H9 K9 J9) -R(N9,H9,K9,SUBT(J9,1),J9) v Q(N9,H9,K9,J9)
10. (N10 H10 K10) R(N10,H10,K10,H10,K10)
```

[仕 様 1]

但，入力のA[0]... A[N-1]はside effectとして与えることにする。
L(N,ろ)　：ろはA[0]… A[N-1]に含まれるupsequenceの最大の長さである。
P(N,ろ,l,k)：ろ,Nについてはしと同じ。かつ，kは長さlのupsequenceのminimum

right-most element である。

Q(N,h,k,j): M[h]≤A[n]<M[k] が成り立つとき，jは，M[j-1]≤A[n]<M[j]を満すうである。

R(N,h,k,i,j): 〃 ，jは，M[i-1]≤A[n]<M[j]を満すうである。

PilotF1には仕様はformulaで与えた。この例題は前処理済みの仕様を示している。各記号の宣言に関する部分は略した。また，サブルーチン(関数)名は合成中に与えた。次に合成されたプログラムを示す。

```
(LUPBCF (LAMBDA (NN) (PROG (Z M4 M3 N11 K11 H11) (SETQ M1 (ARRAY 100 NIL 0)) (
SETQ N11 1) (SETQ K11 1) (SETQ H11 1) (SETA M1 1 (A 0)) label1P (COND ((EQ N11 NN
) (SETQ M3 NN) (SETQ M4 K11) (GO label1L)) ((AND (ILEQ (ELT M1 K11) (A N11)) (NEQ
 N11 NN)) (SETA M1 (ADD1 K11) (A N11)) (SETQ H11 (ADD1 K11)) (SETQ K11 (ADD1 K11
)) (SETQ N11 (ADD1 N11))) ((AND (ILESSP (A N11) (ELT M1 1)) (NEQ N11 NN)) (SETA
M1 1 (A N11)) (SETQ H11 1) (SETQ N11 (ADD1 N11))) ((AND (ILEQ (ELT M1 1) (A N11)
) (NEQ N11 NN) (ILESSP (A N11) (ELT M1 K11))) (SETQ J (BSRCH NN N11 1 K11)) (
SETA M1 J (A N11)) (SETQ N11 (ADD1 N11))) (T (SETQ Z (QUOTE undefined-at-1)) (GO
 EXIT))) (GO label1P) label1L (COND ((EQ M3 NN) (SETQ Z M4) (GO EXIT)) (T (SETQ Z
(QUOTE undefined-at-2)) (GO EXIT))) (GO label1L) EXIT (RETURN Z))))
```

[プログラム 1 (×イン)]

```
(BSRCH (LAMBDA (NN N20 H20 K20) (PROG (J20 M10 M9 M8 M7 M6 M5 M4 M3 M2) (M2_N20)
 (M3_H20) (M4_K20) (M5_H20) (M6_K20) label1R (if M5~=M6-1 and (ELT M1 (M5+ (M6+1)
) /2) st (A M2) then M6_ (M5+ (M6+1)) /2 elseif M5~=M6-1 and (ILEQ (ELT M1 (M5+
(M6+1)) /2) (A M2)) then M5_ (M5+ (M6+1)) /2 elseif M6-1=M5 then M7_M2 M8_M3
M9_M4 M10_M6 (GO label1Q) else J20_'undefined-at-1 (GO EXIT)) (GO label1R) label1Q
(if N20=M7 and H20=M8 and K20=M9 then J20_M10 (GO EXIT) else J20_'undefined-at-2
 (GO EXIT)) (GO label1Q) EXIT (RETURN J20))))
```

[プログラム 1 (サブ)]

2.3 仕様の更新による等価変換 (1)

次に，binary searchを行うopenサブルーチンを持つようなプログラムへ変換するために，仕様1のAxioms 6～10を次のものと置きかえる。

```
6. (N6 K6 H6) -<=(!M1[1],A(N6)) v -P(N6,K6,H6,!M1[H6]) v R(N6,1,K6,K6) v =(N6,NN
) v -<(A(N6),!M1[K6])
7. (N7 J7 K7) -R(N7,SUBT(J7,1),J7,K7) v P(ADD(N7,1),K7,J7,A(N7))
8. (N8 I8 J8 K8) R(N8,I8,DIV(ADD(I8,ADD(J8,1)),2),K8) v -R(N8,I8,J8,K8) v ->(!M1
[DIV(ADD(I8,ADD(J8,1)),2)],A(N8))
9. (N9 I9 J9 K9) R(N9,DIV(ADD(I9,ADD(J9,1)),2),J9,K9) v -R(N9,I9,J9,K9) v -<=(
!M1[DIV(ADD(I9,ADD(J9,1)),2)],A(N9))
```

[仕様 2 (更新部分)]

仕様2から合成されたプログラムを示す。

```
(LUPBOF (LAMBDA (NN) (PROG (Z M8 M7 N10 Z3 H7 M5 M4 M3 M2) (M1 _ (ARRAY 100 NIL
0)) (N10 _ 1) (Z3 _ 1) (H7 _ 1) ((ELT M1 1) _ (A 0)) label1P (if (N10 = NN) then
(M7 _ NN) (M8 _ Z3) (GO label1L) elseif (N10 ~= NN) then ((ELT M1 (Z3 + 1)) _ (A
N10)) (H7 _ (Z3 + 1)) (Z3 _ (Z3 + 1)) (N10 _ (N10 + 1)) elseif (N10 ~= NN) then
((ELT M1 1) _ (A N10)) (H7 _ 1) (N10 _ (N10 + 1)) elseif (N10 ~= NN) then (M2 _
N10) (M3 _ 1) (M4 _ Z3) (M5 _ Z3) (GO label1R) else (Z _ (QUOTE undefined-at-1))
(GO EXIT)) (GO label1P) label1L (if (M7 = NN) then (Z _ M8) (GO EXIT) else (Z _ (
QUOTE undefined-at-2)) (GO EXIT)) (GO label1L) label1R (if ((M4 - 1) = M3) then (
N10 _ (M2 + 1)) (Z3 _ M5) (H7 _ M4) ((ELT M1 M4) _ (A M2)) (GO label1P)) label1R (
M4 _ ((M3 + (M4 + 1)) / 2)) label1R (M3 _ ((M3 + (M4 + 1)) / 2)) (GO label1R) EXIT
 (RETURN Z))))
```

[プログラム 2]

2.4 仕様の更新による等価変換 (2)

linear search を行う openサブルーチンを持つプログラムへ変換するために，仕様1の Axioms 6～10を次のものと置き換える。合成されたプログラムも示す。

```
6. (N6 K6 H6) -<=(!M1(1),A(N6)) v -P(N6,K6,H6,!M1(H6)) v R(N6,1,K6) v =(N6,NN)
  v -<(A(N6),!M1(K6))
7. (N7 J7 K7) R(N7,ADD(J7,1),K7) v -R(N7,J7,K7) v -<(!M1(J7),A(N7))
8. (N8 J8 K8) P(ADD(N8,1),K8,J8,A(N8)) v -R(N8,J8,K8) v ->=(!M1(J8),A(N8))
```

[仕 様 3 (更新部分)]

```
(LUPLOF (LAMBDA (NN) (PROG (Z M7 M6 N9 K9 H7 M4 M3 M2) (M1 _ (ARRAY 100 NIL 0))
(N9 _ 1) (K9 _ 1) (H7 _ 1) ((ELT M1 1) _ (A 0)) labelP (if (N9 = NN) then (M6 _
NN) (M7 _ K9) (GO labelL) elseif ((ELT M1 K9) LE (A N9)) and (N9 ~= NN) then ((
ELT M1 (K9 + 1)) _ (A N9)) (H7 _ (K9 + 1)) (K9 _ (K9 + 1)) (N9 _ (N9 + 1))
elseif ((A N9) LT (ELT M1 1)) and (N9 ~= NN) then ((ELT M1 1) _ (A N9)) (H7 _ 1)
 (N9 _ (N9 + 1)) elseif ((ELT M1 1) LE (A N9)) and (N9 ~= NN) and ((A N9) LT (
ELT M1 K9)) then (M2 _ N9) (M3 _ 1) (M4 _ K9) (GO labelR) else (Z _ (QUOTE
undefined-at-1)) (GO EXIT)) (GO labelP) labelL (if (M6 = NN) then (Z _ M7) (GO
EXIT) else (Z _ (QUOTE undefined-at-2)) (GO EXIT)) (GO labelL) labelR (if ((ELT
M1 M3) LT (A M2)) then (M3 _ (M3 + 1)) elseif ((ELT M1 M3) GE (A M2)) then (N9 _
 (M2 + 1)) (K9 _ M4) (H7 _ M3) ((ELT M1 M3) _ (A M2)) (GO labelP) else (Z _ (
QUOTE undefined-at-3)) (GO EXIT)) (GO labelR) EXIT (RETURN Z))))
```

[プログラム 3]

2.5 仕様の更新による等価変換 (3)

必要最小範囲の binary searchを行うopenサブルーチンを持つプログラムへ変換するため，仕様2の Axioms 5,6を次のものと置き換える。

```
5. (N5 K5 H5) -<(A(N5),!M1[MAX(1,SUBT(K5,SUBT(NN,N5)))]) v -P(N5,K5,H5,!M1[H5])
  v =(N5,NN) v P(ADD(N5,1),K5,MAX(1,SUBT(K5,SUBT(NN,N5))),A(N5))
6. (N6 K6 H6) -<=(!M1[MAX(1,SUBT(K6,SUBT(NN,N6)))],A(N6)) v -P(N6,K6,H6,!M1[H6])
  v R(N6,MAX(1,SUBT(K6,SUBT(NN,N6))),K6,K6) v =(N6,NN) v -<(A(N6),!M1[K6])
```

[仕 様 4 (更新部分)]

```
(LUPBHF (LAMBDA (NN) (PROG (Z M8 M7 N10 K10 H7 M5 M4 M3 M2) (M1 _ (ARRAY 100 NIL
 0)) (N10 _ 1) (K10 _ 1) (H7 _ 1) ((ELT M1 1) _ (A 0)) labelP (if (N10 = NN)
then (M7 _ NN) (M8 _ K10) (GO labelL) elseif ((ELT M1 K10) LE (A N10)) and (N10
~= NN) then ((ELT M1 (K10 + 1)) _ (A N10)) (H7 _ (K10 + 1)) (K10 _ (K10 + 1)) (
N10 _ (N10 + 1)) elseif ((A N10) LT (ELT M1 (MAX 1 (K10 - (NN - N10))))) and (
N10 ~= NN) then (H7 _ (MAX 1 (K10 - (NN - N10)))) ((ELT M1 (MAX 1 (K10 - (NN -
N10)))) _ (A N10)) (N10 _ (N10 + 1)) elseif ((ELT M1 (MAX 1 (K10 - (NN - N10))))
 LE (A N10)) and (N10 ~= NN) and ((A N10) LT (ELT M1 K10)) then (M2 _ N10) (M3 _
 (MAX 1 (K10 - (NN - N10)))) (M4 _ K10) (M5 _ K10) (GO labelR) else (Z _ (QUOTE
undefined-at-1)) (GO EXIT)) (GO labelP) labelL (if (M7 = NN) then (Z _ M8) (GO
EXIT) else (Z _ (QUOTE undefined-at-2)) (GO EXIT)) (GO labelL) labelR (if ((M4 -
 1) = M3) then (N10 _ (M2 + 1)) (K10 _ M5) (H7 _ M4) ((ELT M1 M4) _ (A M2)) (GO
labelP) elseif ((ELT M1 ((M3 + (M4 + 1)) / 2)) GT (A M2)) then (M4 _ ((M3 + (M4
 + 1)) / 2)) elseif ((ELT M1 ((M3 + (M4 + 1)) / 2)) LE (A M2)) then (M3 _ ((M3 +
(M4 + 1)) / 2)) else (Z _ (QUOTE undefined-at-3)) (GO EXIT)) (GO labelR) EXIT (
```

[プログラム 4]

2.6 仕様の更新による等価変換 (4)

最小個数の配列を使う，binarysearch を行う openサブルーチンを持つプログラムへ変換するため，仕様2の Axiom 3を次のものと置き換え，Axioms 全体に対して，Aを!A1で，!M1を!A1で置き換える。"!"は変数名をロケーション名に変える記号である。

```
3. () P(1,0,1,!A1(1))
```

[仕様 5 (変更部分)]

```
(LUPAOF (LAMBDA (NN) (PROG (Z M7 M6 N10 Z3 H7 M5 M4 M3 M2) (A1 _ (ARRAY 100 NIL
0)) (N10 _ 1) (Z3 _ 0) (H7 _ 1) labelP (if (N10 = NN) then (M6 _ NN) (M7 _ Z3) (
GO labelL) elseif ((ELT A1 Z3) LE (ELT A1 N10)) and (N10 ~= NN) then ((ELT A1 (
Z3 + 1)) _ (ELT A1 N10)) (H7 _ (Z3 + 1)) (Z3 _ (Z3 + 1)) (N10 _ (N10 + 1))
elseif ((ELT A1 N10) LT (ELT A1 1)) and (N10 ~= NN) then ((ELT A1 1) _ (ELT A1
N10)) (H7 _ 1) (N10 _ (N10 + 1)) elseif ((ELT A1 1) LE (ELT A1 N10)) and (N10 ~=
 NN) and ((ELT A1 N10) LT (ELT A1 Z3)) then (M2 _ N10) (M3 _ 1) (M4 _ Z3) (M5 _
Z3) (GO labelR) else (Z _ (QUOTE undefined-at-1)) (GO EXIT)) (GO labelP) labelL
(if (M6 = NN) then (Z _ M7) (GO EXIT) else (Z _ (QUOTE undefined-at-2)) (GO EXIT
)) (GO labelL) labelR (if ((M4 - 1) = M3) then (N10 _ (M2 + 1)) (Z3 _ M5) (H7 _
M4) ((ELT A1 M4) _ (ELT A1 M2)) (GO labelP) elseif ((ELT A1 ((M3 + (M4 + 1)) / 2
)) GT (ELT A1 M2)) then (M4 _ ((M3 + (M4 + 1)) / 2)) elseif ((ELT A1 ((M3 + (M4
+ 1)) / 2)) LE (ELT A1 M2)) then (M3 _ ((M3 + (M4 + 1)) / 2)) else (Z _ (QUOTE
undefined-at-3)) (GO EXIT)) (GO labelR) EXIT (RETURN Z))))
```

[プログラム 5]

2.7 仕様の変更による等価変換 (5)

最少個数の配列を使い，必要最少範囲の binary search を行う open サブルーチンを持つプログラムへ変換するため，仕様4に等価変換(4)と同じ変換を施こす。

```
<<  Specification  >>

~Conjecture :

-1. (NN Z) -L(NN,Z)

Axioms :

2. (Z2 H2) -P(NN,Z2,H2,!A1(H2)) v L(NN,Z2)
3. () P(1,0,1,!A1(1))
4. (N4 K4 H4) -<=(!A1(K4),!A1(N4)) v -P(N4,K4,H4,!A1(H4)) v =(N4,NN) v P(ADD(N4,
1),ADD(K4,1),ADD(K4,1),!A1(N4))
5. (N5 K5 H5) -<(!A1(N5),!A1(MAX(1,SUBT(K5,SUBT(NN,N5))))) v -P(N5,K5,H5,!A1(H5)
) v =(N5,NN) v P(ADD(N5,1),K5,MAX(1,SUBT(K5,SUBT(NN,N5))),!A1(N5))
6. (N6 K6 H6) -<=(!A1(MAX(1,SUBT(K6,SUBT(NN,N6)))),!A1(N6)) v -P(N6,K6,H6,!A1(H6
)) v R(N6,MAX(1,SUBT(K6,SUBT(NN,N6))),K6,K6) v =(N6,NN) v -<(!A1(N6),!A1(K6))
7. (N7 J7 K7) -R(N7,SUBT(J7,1),J7,K7) v P(ADD(N7,1),K7,J7,!A1(N7))
8. (N8 I8 J8 K8) R(N8,I8,DIV(ADD(I8,ADD(J8,1)),2),K8) v -R(N8,I8,J8,K8) v ->(!A1
(DIV(ADD(I8,ADD(J8,1)),2)),!A1(N8))
9. (N9 I9 J9 K9) R(N9,DIV(ADD(I9,ADD(J9,1)),2),J9,K9) v -R(N9,I9,J9,K9) v -<=(
!A1(DIV(ADD(I9,ADD(J9,1)),2)),!A1(N9))
```

[仕様 6]

```
(LUPAHF (LAMBDA (NN) (PROG (Z M7 M6 N10 K10 H7 M5 M4 M3 M2) (A1 _ (ARRAY 100 NIL
0)) (N10 _ 1) (K10 _ 0) (H7 _ 1) labelP (if (N10 = NN) then (M6 _ NN) (M7 _ K10
) (GO labelL) elseif ((ELT A1 K10) LE (ELT A1 N10)) and (N10 ~= NN) then ((ELT
A1 (K10 + 1)) _ (ELT A1 N10)) (H7 _ (K10 + 1)) (K10 _ (K10 + 1)) (N10 _ (N10 + 1
)) elseif ((ELT A1 N10) LT (ELT A1 (MAX 1 (K10 - (NN - N10))))) and (N10 ~= NN)
then (H7 _ (MAX 1 (K10 - (NN - N10)))) ((ELT A1 (MAX 1 (K10 - (NN - N10)))) _ (
ELT A1 N10)) (N10 _ (N10 + 1)) elseif ((ELT A1 (MAX 1 (K10 - (NN - N10)))) LE (
ELT A1 N10)) and (N10 ~= NN) and ((ELT A1 N10) LT (ELT A1 K10)) then (M2 _ N10)
(M3 _ (MAX 1 (K10 - (NN - N10)))) (M4 _ K10) (M5 _ K10) (GO labelR) else (Z _ (
QUOTE undefined-at-1)) (GO EXIT)) (GO labelP) labelL (if (M6 = NN) then (Z _ M7)
 (GO EXIT) else (Z _ (QUOTE undefined-at-2)) (GO EXIT)) (GO labelL) labelR (if (
(M4 - 1) = M3) then (N10 _ (M2 + 1)) (K10 _ M5) (H7 _ M4) ((ELT A1 M4) _ (ELT A1
 M2)) (GO labelP) elseif ((ELT A1 ((M3 + (M4 + 1)) / 2)) GT (ELT A1 M2)) then (
M4 _ ((M3 + (M4 + 1)) / 2)) elseif ((ELT A1 ((M3 + (M4 + 1)) / 2)) LE (ELT A1 M2
)) then (M3 _ ((M3 + (M4 + 1)) / 2)) else (Z _ (QUOTE undefined-at-3)) (GO EXIT)
) (GO labelR) EXIT (RETURN Z))))
```

[プログラム 6]

プログラム6のフローチャートを図2に示す。

3. おわりに

自動合成系LPSのあるプログラミング環境における1開発例を見てもらったが，LPS pilot system (DEC system20. 2060, INTERLISP上)で，各仕様からプログラムへの変換時間を次にあげる。これはCPUタイムで単位は「秒」である。

仕様	CPU time**	Real time	consの数
1*	12.736	146.202	40548
2	9.056	80.753	31139
3	7.129	63.398	20029
4	9.052	90.477	24472
5	9.557	90.596	24700
6	4.518	19.138	34394

(単位=秒)

注 *… メインとサブのそれぞれを合計した.
**… CPU timeはreal timeに影響をうけるらしく あまり信用できない。

紙数の制限から，LPSシステムや例題仕様の説明が不十分になり，プログラムをpretty printで載せられなかったのが残念である。

謝辞　上村義明教授および琴野実君，植田浩市君，柳正栄君，さらにLPSシステムの開発メンバー諸氏に感謝します。

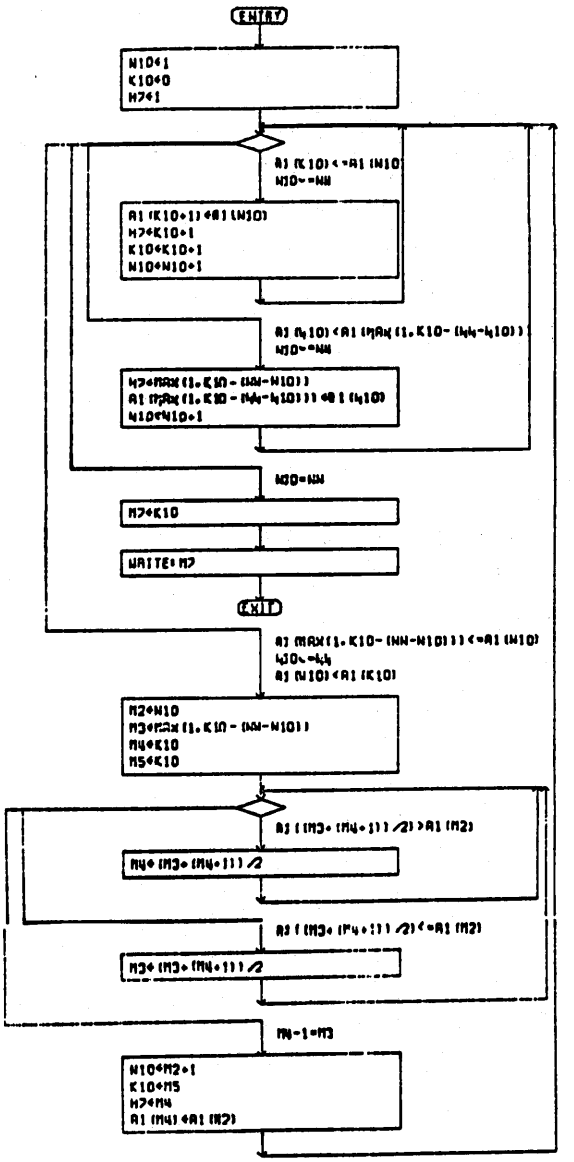

図2　最少個数の配列を使い，必要最少範囲のbinary searchを行うサブルーチンを持つ，LONGEST UPSEQUENCEのフローチャート

参考文献

1. E.W.Dijkstra: Some Beautiful Arguments Using Mathematical Induction. Acta Informatica 13, 1-8 (1980)
2. 謝(1980): LPS pilot systemによる宿題(7A, 8A) プログラムの自動合成。情報処理 PSシンポジウム報告集 pp.160～168, 187～189
3. 謝(1979): プログラミングの形式化とソフトウェア・ツールのありかた。情報処理 vol.20 no.6 pp.481～486
4. 謝(1979): 論理的プログラム合成と構造反証原理。情報処理 SE研究会 9-7
5. 謝(1977): Foundations of Logical Program Synthesis. 情報処理 SE研究会(3月)
6. 謝(1975): Mechanical Flowchart Synthesisについて。信学会 AL75-2