不完全指定順序回路の内部状態数最小化のためのLISP言語プログラム

後　藤　公　雄

LISP Language Program for Minimization of Numbers of Internal States in Incompletely Specified Sequential Machines

Kimio GOTO

Abstract

In this paper, the program using LISP language to realize the minimization of numbers of internal states in the incompletely specified sequential machines in accordance with simpler algorithm, that is, GT one which improves several complex difficulties, is proposed. This algorithm especially has changed and improved the contents of three theorems of Rao & Biswas and how to use them. The LISP language program implemented to this new algorithm was drawn up in accordance with μ LISP-86, and run on the MS-DOS in personal computer PC-9801, and the results of run were compared with those in the BASIC language program for the completely same algorithm. As a result, the running speed for obtaining solutions in this LISP language program was proven to be superior to the BASIC one by 5 to 10 times.

1. 緒　　言

不完全指定順序回路の内部状態数の最小化については古くから検討されており[1),2),3),4)]，最近ではRao & Biswasの方法[5)]が知られている。筆者もこれまで種々の検討を行ってきたが，最近Rao & Biswasの方法の複雑さを改善する方法としてGT法を開発し報告した[9)]。本論文ではこのGT法のアルゴリズムを用いて実際に作成したLISP言語プログラムについて述べる。このプログラムの演算速度はBASIC言語によるものよりも5～10倍まで改善されている。

2. 内部状態数の最小化のアルゴリズム

図1のフローチャートに従いGT法のアルゴリズムの概要を述べる。

［手順1］　不完全指定順序回路の状態遷移表上の異なる2つの現在の内部状態を比較して遷移先内部状態対を探し出し，出力両立するものと，しないものを区別してImplication Table（以後IMPTと呼ぶ）に記入し，両立性対（以後CPと呼ぶ）を求める。

［手順2］　CPの集合から最大両立性クラス（以後MCと呼ぶ）を求める。MCはそれを構成する全ての内部状態対がCPであって他の如何なる両立性クラスにもカバーされないものを言う。

［手順3］　MCの全ての部分集合であるPrimary Compatible Class（以後PCと呼ぶ）と，そのPCのインプライする全てのCPからなる閉包対集合（以後EPと呼ぶ）を求める。

［手順4］　IMPTより求まったCP同士をそのインプリケーション関係を表す矢印線で結んだImplication chain（以後IMPCと呼ぶ）を作成する。

［手順5］　IMPC上で他からインプライされずに他をインプライしているCP，すなわちUCPを調べ，その遷移先のEP中に自分の内部状態を含むCPがあれば，このUCPを除去し，この結果新たに発生したUCPを同じ要領で除去して行き，このような除去ができなくなるまでこれを続ける。このようにして最後に残ったIMPCを基本インプリケーション・チェーン（Basic Implication Chain）と呼び（以後略して

昭和63年9月28日受理

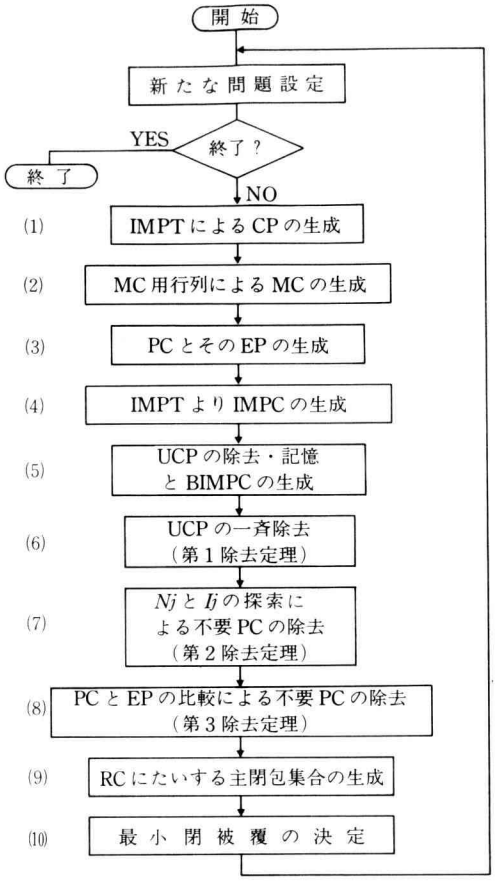

図1.　フローチャート
Fig. 1.　Flow chart

BIMPCと呼ぶ），また，ここで除去されるUCPを弱結合UCPと呼ぶ。

［手順6］　IMPCを作成するときに除去されたUCPを記憶しておき，PCの中から一挙に除去する。これは第一除去定理[9]による。

［手順7］　3個以上の内部状態より成るPC（C_i）についてその各内部状態がそのPCのインプライするEPに含まれているか否かを調べ，含まれない内部状態があればその集合をN_jとする。つぎにこのPCに含まれるCPで，BIMPCの内部でインプライされているCP，すなわちICPに等しくなるもののいくつかの合併集合をI_jとし，

$$C_j = N_j \cup I_j \subset C_i \qquad (1)$$

表1　不完全指定順序回路の状態遷移表の一例
Table 1　An example of state transition table of incompletely specified sequential machine

	X_0		X_1		X_2	
1	3	0	—	—	2	—
2	—	—	4	0	6	—
3	5	1	—	—	—	0
4	—	—	1	1	1	—
5	1	—	—	—	6	—
6	4	—	5	—	6	—

であれば，このC_iは除去する。これは第二除去定理[9]による。

［手順8］　あるPC（C_i）が他のPC（C_j）の内部状態をすべて含んでおり，かつC_iの閉包対集合EP_iがC_jの閉包対集合EP_jに完全にカバーされるとき，すなわち

$$C_i \supset C_j \quad \text{かつ} \quad EP_i \leq EP_j \qquad (2)$$

の場合，C_jを除去する。これは第三除去定理[9]による。これらが全て完了した時点で残ったPCを代表的両立性クラス（Representative Compatible Class）と呼ぶ（以後略してRCと呼ぶ）。

［手順9］　得られた各RCのインプライするEPの各要素を，それを含むそれぞれのRCの集合に置換し，このようなRCの集合とインプライ源のRCとの間で直積を求め，この直積の各要素をインプライ源のRCに関する主閉包集合と呼ぶ。ついで全ての主閉包集合と全ての元の内部状態の間で被覆関係を調べ，最小被覆の組を求める。

［手順10］　この結果残った主閉包集合の組で閉じているものを採用する。これが最小閉被覆解となる。

つぎに簡単な事例を用いて上のアルゴリズムに従った計算を述べる。表1はある不完全指定順序回路の状態遷移表である。この状態遷移表にたいしてIMPTを示すと図2が得られる。この結果，CPの集合として

$$\{56, 14, 12, 26, 16, 34, 45, 36, 23, 25\} \qquad (3)$$

が得られる。これは手順1にしたがったものである。ついで手順2として式(3)のCPの集合からMCの集合を求めると，

$$\{126, 236, 256, 14, 34, 45\}. \qquad (4)$$

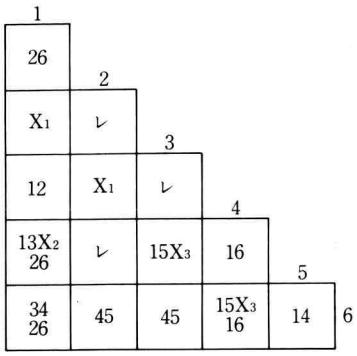

図2　表1の機械にたいするIMPT
Fig. 2.　Implication Table for the machine of Table 1

表2　表1の機械にたいするPCとEP
Table 2　PCs and Those EPs for the machine of Table 1

PC	EP
× 12	26 45 16 34 ……D 1
× 14	12 26 45 16 34 ……D 1
←16	34 26 45 ……D 3
23	φ
25	φ
←26	45 16 34 ……D 3
34	φ
× 36	45 16 34 26 ……D 1
45	16 34 26
× 56	14 12 26 45 16 34 ……D 1
→126	45 34
× 236	45 16 34 ……D 2
× 256	45 16 34 14 12 ……D 2

手順3として，式(4)の部分集合を求め，PCの集合

$$\{12, 14, 16, 23, 25, 26, 34, 36, 45, 56, 126, 236, 256\} \quad (5)$$

が求まり，式(5)の各要素，すなわちPCのインプライするEPを求めると，表2のようになる。さらに，手順4として手順1で求めた式(3)の要素となっているCP相互間のインプリケーションの関係より図3に示すIMPCが得られる。手順5として図3のIMPC上で不要なUCPとして，56, 36, 14および12を順次除去用として記憶して行き，最後に一点鎖線で囲む

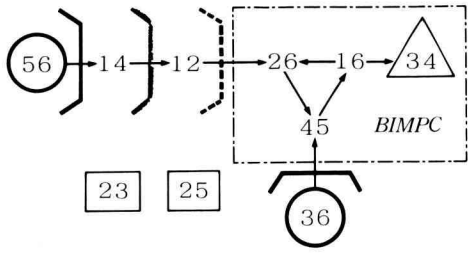

図3　表1の機械にたいするIMPC
Fig. 3.　Implication chain for the machine of Table 1

BIMPCが得られる。手順6として手順5で記憶してある不要UCPを一挙に除去する。このように除去されるPCを表2で×印とD1（除去定理1の意味）で示す。手順7として，表2の3個以上の内部状態を含むPCの中で$C_i=236$に注目すると，26はBIMPCに含まれ，2はEPの要素に含まれないから，

$$I_j=26, \quad N_j=2 \quad (6)$$

であり，式(1)より

$$C_j=26 \subset C_i=236 \quad (7)$$

が成立し，除去定理2より236は除去される。同様に256も除去される。これを表2で×印とD2で示す。手順8として，表2で残ったPC

$$\{16, 23, 25, 26, 34, 45, 126\} \quad (8)$$

に注目すると，たとえば，

$$16 \subset 126 \quad (9)$$

であり，それぞれのインプライするEPの関係は，

$$\{34, 26, 45\} > \{45, 34\} \quad (10)$$

であるから，16は除去される。同様に26も除去される。これを表2で×印とD3（除去定理3の意味）で示す。残ったPC，すなわちRC

$$\{23, 25, 34, 45, 126\} \quad (11)$$

について，

$$23=C_1,\ 25=C_2,\ 34=C_3,\ 45=C_4,\ 126=C_5 \quad (12)$$

として，手順9の主閉包集合を求めると，表3が得られる。さらに，表3より$C_4C_3C_5$は最小閉被覆であることがわかる。この例は，手順9と10については簡単な事例となっているが，実際にはもっと複雑になること

表3　EPのRCによる表現と主閉包集合
Table 3　Method to express EPs by using RC and each prime closure set for each RC

RC	EP	主閉包集合
23	ϕ	C_1
25	ϕ	C_2
34	ϕ	C_3
45	16　34　26 → 126	C_4　C_3　C_5
126	45　34	C_5　C_4　C_3

注）　23 ⇨ C_1, 25 ⇨ C_2, 34 ⇨ C_3, 126 ⇨ C_5

が多い。

3.　プログラム化手法

上述したアルゴリズムを用いてLISP言語プログラムを作成した。サブルーチン用の関数は約50個となった。またプログラム作成にあたってはμLISP-86の組込み関数を使用した。

つぎにプログラムの作成上配慮した点について述べる。

(1)　IMPTの生成

IMPT-CREATE関数によりIMPTを生成する。この関数では，まずIMPTの各セル（CPに対応）の中で最初に出力非両立となるものをIMPT-X1F関数によって求める。IMPT-X1F関数は状態遷移表（以後STAT-TBLと呼ぶ）の異なる行間の出力値にたいし排他的論理和LOGXORをとって出力非両立を判定し，IMPTの対応セルにX1を記入する。また，それ以外のセルにインプライ先のCPを記入するためIMPT-ELSE関数を用いる。さらにIMPTの各セルに記入されたCPの中で非両立となるものにたいし，X_n(ここで$n \geq 2$)を記入するため，IMPT-X_n関数を用いる。この関数は，IMPTのあるセルにX_nを記入すると，それによってさらに他のセルにX_{n+1}を記入することになるので再帰関数として使用される。これらの関数ではIMPT内部やSTAT-TBL内部のセル内容を調べたり，その内容を書き換えたりするので，AREF, F1ND, UNPACK, PACK ＊, SORT, PUSHなどの組込み関数が多用される。なお，IMPTの各セルの内容から，CPの各要素を格納するCC-TBLが求まる。

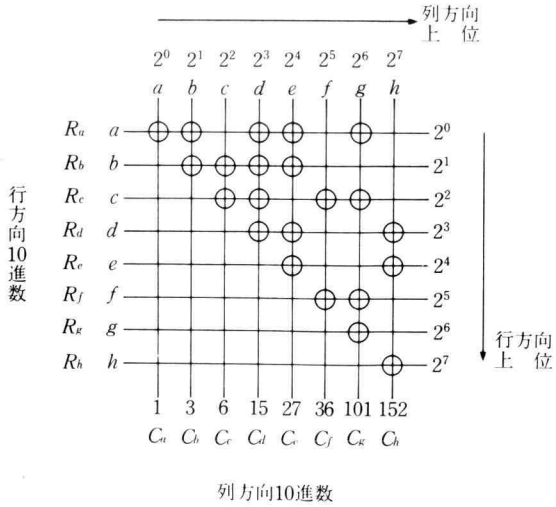

図4　MC用行列の一例
Fig. 4.　An example of MC matrix

(2)　MCとPCの生成

図4に示すような内部状態に対応する各行線（たとえばa, b, c, d, e, f, g, h）と各列線（たとえばa, b, c, d, e, f, g, h）より成り，しかも右上半分のみに○印または数字が記入されたMC用行列がある。この行列では，CPがijで与えられたとき，i行i列の交点，i行j列の交点，およびj行j列の交点に○印または数字が記入されている。すべてのCPがこのようにしてMC用行列に記入されているものとし，この行列からMCを生成する。

いま，行線と列線の交点上の○印を1，○印のない交点を0と見なし，最上行の行線をLSB，最下行の行線をMSBとし，最上行から最下行に向かって順に上位のビット値を表すものとし，各列線ごとに0，1で構成された2進数を10進数に変換する。また，図4の各行上で，最左列がLSB，順に右の列に移るにしたがい上位ビットとなり，最右列がMSBとなるものと考え，最上行を10進数で表してみる。このようにして得られた図4の各列線$a, b, \cdots, h$を表す10進数をそれぞれ$C_a, C_b, \cdots, C_h$とし，各行線$a, b, \cdots, h$を表す10進数を，それぞれ$R_a, R_b, \cdots, R_h$とする。図4で，たとえば，C_aとC_bの間，およびR_aとC_aの間でLOGANDをとると，

$$C_a \text{ logand } C_b = 1 \text{ logand } 3 = 1, \quad (13a)$$
$$R_a \text{ logand } C_a = 91 \text{ logand } 1 = 1. \quad (13b)$$

同様に C_b と C_d，および R_a と C_b の間でLOGANDをとると，

$$C_b \text{ logand } C_d = 3 \text{ logand } 15 = 3, \quad (14a)$$
$$R_a \text{ logand } C_b = 91 \text{ logand } 3 = 3. \quad (14b)$$

つづいて C_b の代りに C_d，C_d の代りに C_e を入れ換えて同じ操作を繰り返し，結局，

$$C_a \text{ logand } C_b = R_a \text{ logand } C_a, \quad (15a)$$
$$C_b \text{ logand } C_c \neq R_a \text{ logand } C_b, \quad (15b)$$
$$C_b \text{ logand } C_d = R_a \text{ logand } C_b, \quad (15c)$$
$$C_a \text{ logand } C_e = R_a \text{ logand } C_d, \quad (15d)$$
$$C_e \text{ logand } C_x \neq R_a \text{ logand } C_e. \quad (15e)$$

ここで $x = f, g,$ または h.

式(15)の成立より，abdeはMCとなることは明らかである。このようなMCは，すべての行と列について順を追って調べてゆくことにより全部生成でき，MCC-TBLに格納される。また，MCの生成過程で作られるMCの部分集合としてのPCはPC-TBLに格納される。

このアルゴリズムを実現するため，MCC-CREATE, MC-TBL, MC-MATRIX, MC-PC-CREATE, MCC-FILTER-SET, MCC-JUDGEおよびMCC-CORE関数を用いている。MCC-CREATE関数はこれらの関数を呼び出すメイン関数である。MC-TBL関数はMC用行列の列の10進数の格納用であり，MC-MATRIX関数はCC-TBLからCP要素 ij を取り出し，これをMC用行列の (i, j), (i, j) および (j, j) の3つの交点に書き込むため，その都度列線10進数に変換し相互間でLOGIORをとるようにし，得られた列の10進数をMC-TBLに格納する。MCC-FILTER-SET関数は上述した行方向の10進数を生成して，これをELEFILTER変数に格納する。MCC-JUDGE関数は上述したアルゴリズムにしたがって，MC-TBLに格納されている列の10進数相互間のLOGANDをとり，また列の10進数とELEFILTER内の10進数とのLOGANDの比較を行い，PCとなるものを選出し，MCC-CORE関数は真にMCとなるもののみを選出している。このプログラムでは，LOGIOR, LOGANDのほかにCHAR-CODE, SHIFT, CODE-CHAR, SUBSTRING, EVAL, INTEGER-LENGTH, PUSH, PUSHNEW, UNPACKなどの組込み関数が使用される。

(3) IMPC, PCおよびEPの生成

IMPCを作成するため，CC-TBLと同じ長さを持った大局変数としての配列リストIMPCを設定し，CC-TBL内の要素CPの位置と同じIMPC内の位置に，そのCPによってインプライされるCPをIMPTのセル内から抽出して格納する。このため，MAKE-ARRAY, LENGTH, AREF, CHAR-CODE, UNPACKなどの組込み関数を使用する。表1の機械にたいする変数CC-TBLと配列リストIMPC内の要素の対応関係を図5の(a), (b)に示す。

つぎにPC-TBLの各要素としてのPCの位置と同じ位置にEPをリストとして配列する大局変数EP-TBLを生成する。このためEP-TBLはPC-TBLと同じ長さを持つ。PC-TBLとEP-TBLとの一例を図5(c), (d)に示す。各EPの生成はPCの長さが2のものと3以上のものに分けてEP-CREATE関数で行われる。まず，長さ2のPCについては，CC-TBL上のCPを選び，これを変数EP-TMPに格納してサブルーチン関数EP-CREATE-SUBを呼び，IMPCのリストでCC-TBL上のこのCPに対応する位置にあるものを探し，このリストの中のCPでまだEP-TMPに含まれていないものを求めて変数EP-TMPにPUSHする。このようにPUSHされたCPがNULLでなければ，このCPはまだ他をインプライし得るので，再帰的にEP-CREATE-SUB関数へ戻る。このようにして得られたEP-TMPから最初に設定したPCを除去してEP-TBLの配列要素とする。このプログラムには，MAKE-ARRAY, REMOVE, POSITION, AREF, PUSH, LISTなどの組込み関数が用いられる。さらに長さ3以上のPCについては，EP-ELSE関数が使用される。この関数を用いて長さ3以上のPCを構成する長さ2のすべてのPCにたいし，すでに作成されているEP-TBL内のいくつかのEPのリストの合併集合を作り，これを与えられたPCのPC-TBL内の位置に対応するEP-TBL位置に格納する。このプログラムにはREMOVE, REMOVE-IF, UNION, SORT, POSITION, AREFなどの組込み関数が使用される。

(4) BIMPCの生成

すでに生成されたCC-TBL, IMPCおよびEP-TBLを用いてBIMPCと弱結合UCPを生成する。まず，IMPCでUCPを探し，そのUCPを表すPC-TBL中のリスト位置に対応するEP-TBLのリスト内にこのUCPの内部状態を含むCPがあれば，このUCPを

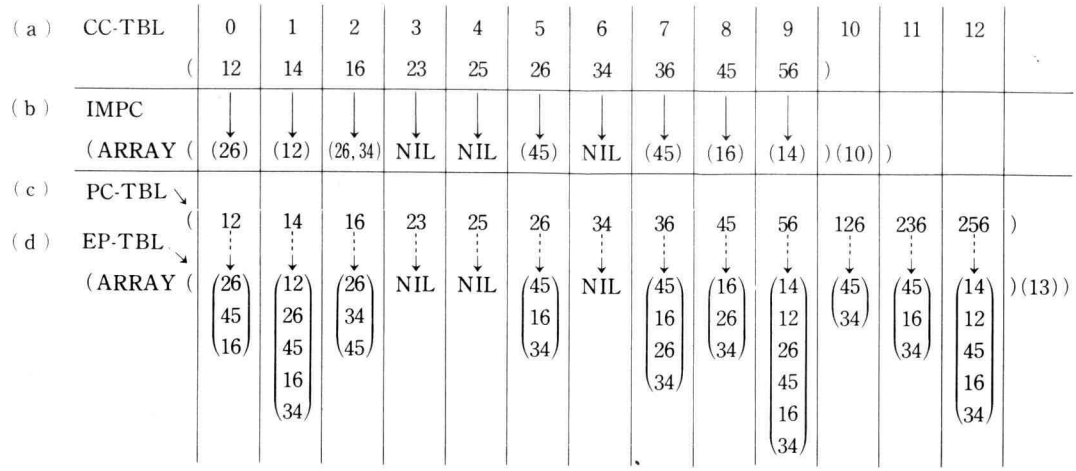

図5.　CC-TBL, IMPC, PC-TBL および EP-TBL の相互関係
Fig. 5.　Interrelations between CC-TBL, IMPC, PC-TBL and EP-TBL

DUCP-TBL 変数に記憶し，IMPC の要素を格納しておいた BIMPC-TBL 変数からこの UCP を除去して BIMPC-TBL の内容を更新し，さらに弱結合 UCP を探す。弱結合 UCP が除去されてしまった最後の BIMPC-TBL の内容は完全な BIMPC の集合となる。

(5)　各除去定理による PC の除去

BIMPC の生成中に変数 DUCP-TBL 内に記憶された弱結合 UCP を PC-TBL から組込み関数 REMOVE を用いて一挙に除去するようにして第一除去定理が実行される。

第二除去定理の適用は，PC の内部状態数が 3 以上の場合を対象とすればよいので，PC-TBL から組込み関数 PINT-LENGTH や REMOVE-IF を用いてこのような PC を ELSE-LIST 変数に格納する。これらの PC について NJ-CREATE 関数を呼んで N_j を求め，これを NJ-TBL に格納する。また，PC$=N_j$ のときには I_j の如何にかかわらず，この PC は除去できないから，組込み関数 UNLESS を用いて PC$\supset N_j$ の場合のみ I_j を選択し，IJ-CREATE 関数によって I_j を求め，IJ-TBL に格納する。さらに組込み関数 UNION を用いて IJ-TBL と NJ-TBL の和集合を求め，その和集合を含む PC を組込み関数 SUBSETP によって確認し，この PC を D2-TBL に記憶し，最後に組込み関数 REMOVE を用いて PC-TBL から除去する。NJ-CREATE 関数は EP-TBL 内の対応するリストに含まれる要素の中にインプライ源の PC の内部状態を含まないか組込み関数 FIND により調べる。IJ-CREATE 関数は，BIMPC-TBL 中の UCP を除去し，残りの要素の中で PC に含まれるものがないか調べる。

第三除去定理の適用にあたっては，PC-TBL 内の残った要素間およびこれらの要素の PC-TBL 上の位置と対応する位置にある EP-TBL 上のリスト相互間の包含関係を組込み関数 SUBSETP を用いて確認することによって行われる。

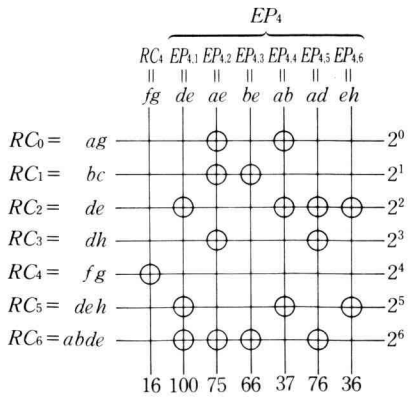

図6.　主閉色集合行列の一例
（$i=4$, $j=1\sim6$, $k=0\sim6$）
Fig. 6.　An example of prime closure set matrix
(for $i=4$, $j=1\sim6$, and $k=0\sim6$)

(6) 最小閉被覆の決定

最後まで残ったPC，すなわちRCにたいする主閉包集合を生成する。この主閉包集合は，RCとそれのインプライするEPの要素CPをカバーするRCの集合との直積によって求まる。ここで，与えられた RC_i にたいするEPの要素CPを表す列線 $EP_{i,j}$ と，その行線 RC_k とよりなる主閉包集合行列を図6に示す。このプログラムでは各列線上の RC_k の存否に対応して1，0を与え，最上行をLBS，最下行をMSBとし，上行から下行に向かってビットの重み付けが上がるものとし

```
Problem No.4

START-TIME=0

        状態遷移表
(ARRAY ((1 (3 0) (- -) (2 -)) (2 (- -) (4 0) (6 -)) (3 (5 1) (- -) (- 0)) (4 (-
-) (1 1) (1 -)) (5 (1 -) (- -) (6 -)) (6 (4 -) (5 -) (6 -))) (6 4))

        IMPLICATION TABLE
(ARRAY (((26) NIL NIL NIL NIL) ((X1) NIL NIL NIL NIL) ((12) (X1) NIL NIL NIL) (
(X2 26 13) NIL (X3 15) (16) NIL) ((26 34) (45) (45) (X3 16 15) (14))) (5 5))

        両立性クラス
(12 14 16 23 25 26 34 36 45 56)

        最大両立性クラス
(45 34 256 236 14 126)

        PRIMALY COMPATIBLE CLASS
(12 14 16 23 25 26 34 36 45 56 126 236 256)
MCC_PC-TIME=200

        IMPLICATION CHAIN
(ARRAY ((26) (12) (26 34) NIL NIL (45) NIL (45) (16) (14)) (10))

        閉包対集合
(ARRAY ((34 16 45 26) (34 16 45 26 12) (45 34 26) NIL NIL (34 16 45) NIL (34 26
16 45) (34 26 16) (34 16 45 26 12 14) (34 45) (34 16 45) (34 16 45 12 14)) (13)
)
EP-TIME=200
((25 23) (16 23 25 26 34 45))
BIMPC-TIME=200

        除去されたUC対　(D1)
(12 14 56 36)
D2
(256 236)
D3
(26 16)

        代表的両立性クラス
(23 25 34 45 126)
RC-TIME=200

        最小被覆の構成要素
((3 2 4) (2) (1) (0))
(1 13 11 3 5 1)
COVER2-TBL
((3 2 4))

        最小状態機械
((45 34 126))

END-TIME=300
```

図7.　実行結果の一例

Fig. 7.　An example of operating result

て2進数を考え，各列のこの値を10進数で表す。図6の例では，$RC_4=6$, $EP_{4,1}=100$, $EP_{4,2}=75$, $EP_{4,3}=66$, $EP_{4,4}=33$, $EP_{4,5}=76$, $EP_{4,6}=36$ となり，これら相互間のLOGANDをとれば，不要な列線は消去される。これは吸収率の適用に相当する。たとえば，

$$EP_{4,1} \text{ logand } EP_{4,6}=100 \text{ logand } 36=36=EP_{4,6}$$
$$EP_{4,2} \text{ logand } EP_{4,3}=75 \text{ logand } 66=66=EP_{4,3}$$
$$EP_{4,4} \text{ logand } EP_{4,6}=37 \text{ logand } 36=36=EP_{4,6}$$

より，列線 $EP_{4,1}$, $EP_{4,2}$, $EP_{4,4}$ は消去される。実際にはこのようにして残った列線相互間で主閉包集合を作り，最小閉被覆を求めている。

4. 演算結果と検討

このプログラムに従って，パーソナル・コンピュータ PC-9801（NEC）を用いてMS-DOS上で実行させた結果をつぎに示す。使用した言語は μ-LISP-86 である。

図7は表1の状態遷移表で示した不完全指定順序回路にたいするものである。

表4　プログラム実行用の不完全指定順序回路の6つの状態遷移表
Table 4　Six state transition tables to incompletely specified sequential circuits for this program running

問1		
	1	3, 0, —, 2, —
	2	—, —, 4, 0, 6, —
	3	5, 1, —, —, —, 0
	4	—, —, 1, 1, 1, —
	5	1, —, —, —, 6, —
	6	4, 0, 5, —, 6, —

問2		
	a	a, 0, —, —, d, 0, e, 1, b, 0, a, —, —, —
	b	b, 0, d, 1, a, —, —, —, a, —, a, 1, —, —
	c	b, 0, d, 1, a, 1, —, —, —, —, —, —, g, 0
	d	—, —, e, —, —, —, b, —, b, 0, —, —, a, —
	e	b, —, e, —, a, —, —, —, b, —, e, —, a, 1
	f	b, 0, c, —, —, 1, h, 1, f, 1, g, 0, —, —
	g	—, —, c, 1, —, —, e, 1, —, —, g, 0, f, 0
	h	a, 1, e, 0, d, 1, b, 0, b, —, e, —, a, 1

問3		
	a	a, 0, —, —, e, —, b, 1
	b	e, —, c, 1, b, —, —, —
	c	—, —, b, 0, —, 1, d, 0
	d	a, 0, —, —, f, 1, b, —
	e	b, 0, —, —, b, 0, —, —
	f	—, —,, 1, —, 0, g, 1
	g	d, 1, d, —, —, —, —, 0

問4		
	1	6, 0, 1, 0, 2, 0, 4, 0
	2	—, —, 6, 0, 2, 0, 4, 0
	3	—, —, 1, 0, 3, 0, 4, 0
	4	5, 1, 0, 0, 2, 0, 4, 0
	5	5, 1, 5, 0, 3, 0, 4, 0
	6	1, 0, 6, 0, 3, 0, 4, 0

問5		
	a	—, —, g, 0, e, 1, d, —
	b	a, —, d, —, —, —, —, 0
	c	c, —, —, 0, —, —, g, 1
	d	e, 0, —, —, a, —, —, —
	e	—, 1, f, —, —, 1, —, 1
	f	—, 1, e, —, a, 1, —, 1
	g	f, —, —, 1, b, —, h, —
	h	c, —, —, —, a, 0, —, —

問6		
	1	—, —, 3, 1, 5, 1, 2, 1
	2	5, 0, —, —, —, —, —, —
	3	6, 0, 6, 1, —, —, —, —
	4	—, —, —, —, 2, 1, —, —
	5	—, —, 6, 0, 1, 0, 4, 1
	6	3, 0, —, —, 2, 0, 3, 1

表5　表4の問題にたいする2つの言語（BASIC, LISP）による演算時間（sec）の比較
Table 5　Comparison of operating times (sec) between two different language programs (BASIC and LISP) for problems in Table 4

問題	言語	除去部まで T_1		被覆部 T_2		計 T_t	
問1	BASIC	59	0.12	10	0.10	69	0.12
	LISP	7		1		8	
問2	BASIC	23	0.17	9	0.11	32	0.16
	LISP	4		1		5	
問3	BASIC	25	0.16	3	0.33	28	0.18
	LISP	4		1		5	
問4	BASIC	28	0.14	3	0.33	31	0.16
	LISP	4		1		5	
問5	BASIC	86	0.09	12	0.08	98	0.09
	LISP	8		1		9	
問6	BASIC	20	0.15	14	0.21	34	0.18
	LISP	3		3		6	

また，表4に示す6つの不完全指定順序回路の状態遷移表（問1～6）に示されるものについて，このプログラムを実行するのに要した演算時間[10]と，全く同じアルゴリズムの下でBASIC言語を用いてプログラムを作成し，同じパーソナル・コンピュータで実行した演算時間とを比較した。この結果を表5に示す。この表には，演算開始からPCにたいする除去定理の適用までに要する演算時間 T_1，それ以後最小閉被覆を求めるまでに要する演算時間 T_2，および総演算時間 T_t が示してある。この表より，つぎのようなことが言える。

(a) 言語がBASICであってもLISPであっても T_1 と T_2 の時間比は同じ問題であればそれほど大差はない。

(b) 総合演算時間 T_t はBASIC言語では問題によって大差を生じ，28 sec～98 secとなり，その比は2.5（＝98/28）となったのにたいし，LISP言語では5 sec～9 secで，その比は1.8（＝9/5）となった。

(c) BASIC言語プログラムに比し，LISP言語プログラムによる演算時間に著しく向上した。LISP演算時間とBASIC演算時間との比は，T_1 については約1/6～1/11となり，T_2 については約1/3～1/13となった。また総合時間 T_t については1/5.5～1/11となり，ほぼ T_1 にたいするものと等しくなった。

以上の結果の中で，(a)の結果は基本アルゴリズムが両方の言語にたいして同じであったため，T_1, T_2 に要する時間比率が変らなかったことを物語っている。(b)の結果はBASIC言語よりもLISP言語がこの種の問題処理にあたって問題の複雑度に影響されにくいこと，換言すればこの種の問題にたいしてはLISP言語の方が適性を持つことを物語っている。さらに(c)の結果に示されるような総合演算時間 T_t の大幅な向上はLISP言語のこの種の問題への適性を物語っている。LISPのように人工知能向け言語として開発された記号処理言語の方が，BASIC言語のような人間向けの言語として開発された数値計算向き高級言語よりも，多くの数の集合を処理するような，すなわち本論文でとり上げられたような問題の計算には適していると言える。

5. 結　　言

以上に述べたように，LISP言語によるプログラムの開発によって不完全指定順序回路の内部状態数の最小化の算出時間が大幅に改善された。なお，この論文では特に触れなかったが，6つの問題以外のさらに複雑な問題についてRCの導出は可能であるが，主閉包集合の導出がメモリ不足のため不可能となるという問題が残された。しかし，この問題も解決ずみであり[12]，いずれそのプログラムについても報告する予定である。

なお，本研究のプログラム作成に尽力された昭和61年度卒業研究生の行冨信増君と飯田正樹君に謝意を表する。また，昭和59年度卒業研究生の佐藤裕康君と昭和60年度の卒業研究生新川和誠君には本研究の基礎を築いていただいた。記して謝意を表する。

参 考 文 献

1) Paul, M.C. and Unger, S.H.: Minimizing the Number of States in Incompletely Specified Sequential Switching Functions, *IRE Trans.*

Electron. Comput., Vol. EC-8, pp. 356-367 (1959).

2) Unger, S.H.: Flow Table Simplification —Some Useful Aids, *IEEE Trans. Electron. Comput.*, Vol. EC-14, pp. 472-475 (1965).

3) McClusky, E.J.: Minimum-state Sequential Circuits for a Restricted Class of Incompletely Specified Flow Tables, *B. S. T. J.*, Vol. 41, pp. 1759-1768 (1962).

4) Biswas, N.N.: State Minimization of Incompletely Specified Sequential Machines, *IEEE Trans. Comput.*, Vol. C-23, pp. 80-84 (1974).

5) Rao, C.V.S. and Biswas, N.N.: Minimization of Incompletely Specified Sequential Machines, *IEEE Trans. Comput.*, Vol. C-24, pp. 1089-1100 (1975).

6) 後藤, 小松: 不完全定義順序回路の分類と内部状態最小化のための基本的手法, 昭和55年度信学会総全大, 1217 (1980).

7) 後藤: 複雑な形の不完全定義順序回路の内部状態最小化の一手法, 昭和55年度信学会総全大, 1216 (1980).

8) 後藤, 日笠山: 両立性対を用いた不完全指定順序回路の最小化手法, 第31回情報処理学会全国大会論文集, 3J-10 (1985).

9) 後藤: IMPCと両立性対の使用による不完全指定順序回路の最小化の一手法, 情報処理学会論文誌, Vol. 28, No. 6, pp. 646-657 (1987).

10) 後藤, 行冨: 不完全指定順序回路の最小化の新手法のLISP言語プログラムによる実現, 昭和62年電気学会全国大会論文集, 1444 (1987).

11) 後藤, 飯田, 高橋: 不完全指定順序回路の最小化のための2つのアルゴリズムの比較, 昭和62年信学会総合全国大会論文集, 310 (1987).

12) 後藤: 主閉包集合の上限値設定による不完全指定順序回路の複数最小解の生成法, 情報処理学会論文誌, Vol. 29, No. 9, pp. 873-887 (1988).