LispマシンALPS/Iの性能評価

井田昌之，小林茂男，山方宏修（青山学院大学）

1．はじめに

ALPS/Iは マイクロコンピュータ8080と64kwのバルクメモリを中心とした専用マシンであり，その目標は実際的なLispプログラムの経済的走行と会話型の専有処理形態の提供にある。51年1月のハードウェア及びLispプロセッサ1版の誕生後，度々の改良の結果現有システムは約100のsubr,fsubrをPROM中に組み込むようになり，また速度も1版に比較して約1.5倍にあがっている。

ここではALPS/Iの機能の概略を紹介し，その実例として箱入娘パズルの解法を取り上げて報告する。

2．Lisp処理系について

ALPS/IのLispプロセッサは，Lisp1.5を基調としてEvalquote入力を解釈実行する約2kbyteのプログラムで，PROMに書き込まれ使用される。関数Evalquoteは文献1と同じ仕様に作られている。apply,evalは若干の高速化のための手直しに加え，付加機能を実現するための手続きが入れられている。evconは再帰的ではなく，ループで実現されている。

Lispシステムとしての ALPS/Iの特徴は，次のようにまとめられる。

①Evalquote インタプリタ without A-list：変数束縛はstack-bind, Funarg問題は束縛凍結により解決

②次のデータタイプを持つ

A．基本整数：アドレスそのものが数値

B．数　　：1wの数（整数及び浮動小数点数（現在組込中））

C．H-分子：H-領域上の属性，値，鍵をもつユニークオブジェクト

C-1．アトミックシンボル：pnameを鍵とする。属性はundef, subr, fsubr, expr, fexpr, apval, hexpr, harray

C-2．ハッシュ化配列要素：配列名aと添字βのドット対を鍵とする。aはアトミックシンボル，βは任意のS式でよい。

C-3．連想計算連想子：HEXPR関数名と実引数のドット対を鍵とする。

C-4．ラベル連想子：ラベル名とプログ定義体のドット対を鍵とする。

D．L-分子：ドット対のリストセル

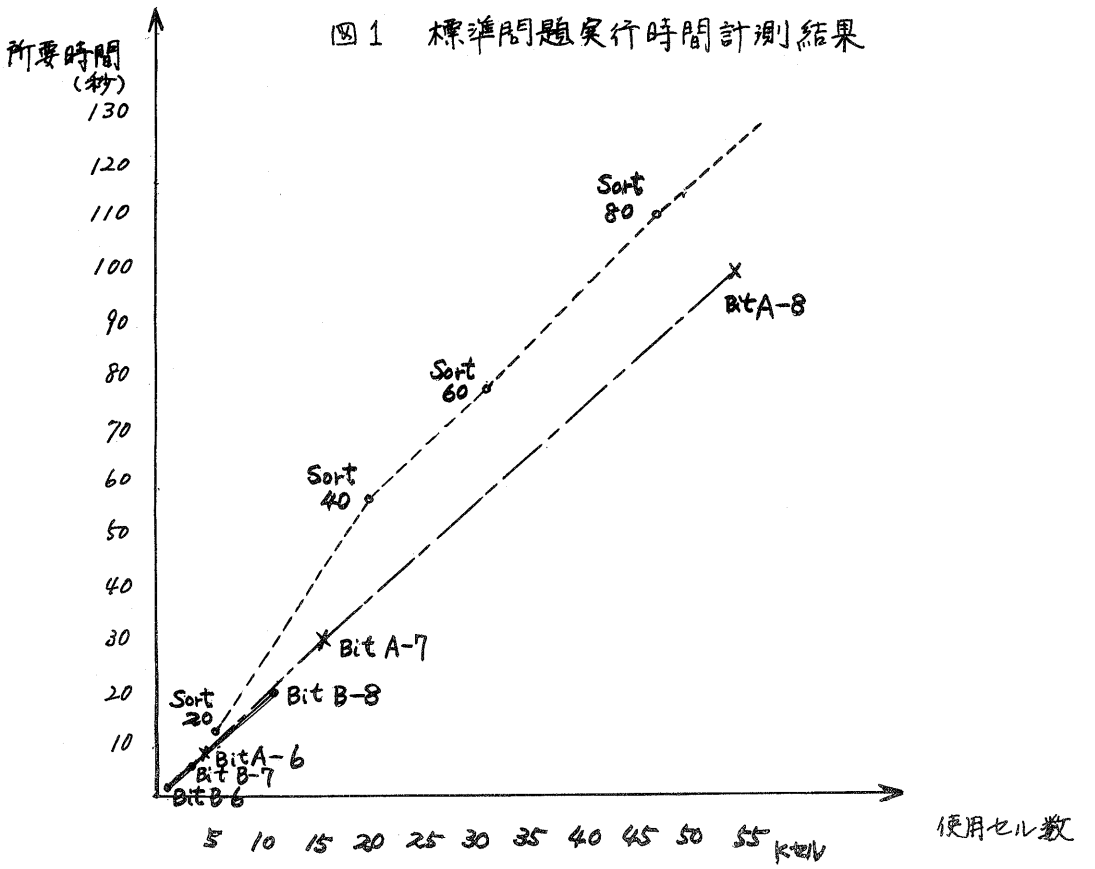
③約100の組込関数を持つTurn keyシステムである。

物理的な上限個数は次の通りである。

数＝2048個（内1024個は基本整数），H-分子＝2048個，L-分子＝57,344個

74年夏のシンポジウムの標準問題の実行結果を図1に示す。Sort-100は137秒を要し，メモリは56kセル以上を使っている。実行時間の計測は11回まわしで f(x) を実行させる汎関数をオブジェクトプログラムロード機能を用いてロードし，実測したものである。（Sortは1回だけ）

図1　標準問題実行時間計測結果



④中間言語ドライブによるプログインタプリタ：最高10個までのプログを同時に保持できるエントリー表と256語の中間言語ロード域（あふれは*PROGNに書き換えられつリーストレージへとCHAINされる）があり，プログはリスト表現からアレイ表現に書き換えられる。そして，アレイ上の各ステートメントを逐次解釈実行する形で処理が進められる。ラベルはH－領域上にラベル連想子を作るのに用いられ，中間言語域上は何も存在しない。

この経過について述べると次のようになる。高速化を計るにはコンパイラが本質的であり，インタプリタと比べて10倍くらい速くなるといわれている。しかしコンパイラの作成に着手することは現段階としては不要であったので，インタプリタの範囲内での簡易的な高速化を考えた。まず目についたのがプログである。COND式による定義やマクロ型の定義にはそれなりの簡潔さがあり，その処理もうなずけるものがあるがprog形式にとっては非常に原始的な方法が今もって行なわれているようである。プログインタプリタの定義には次のようなものがある。（ALPS/Iのプログインタプリタの第1版はこれであった。）

ただし，*setq及び*setは必ず新たに束縛を生成するsetq及びsetとする。*assoc，evalはスタックバインドのためa-list引数を露としたものとする。

```
prog(x) = prog((xx;val;v);
   mapc(car(x);λ((j);*setq(j;NIL)));
   xx:=cdr(x);
   map(xx;λ((j);(atom(car(j))→ *set(car(j);cdr(j)))));
exec-loop:
   (null(xx)→ return(val);
    atom(car(xx))→ go(advance));
   v:=car(xx);
   xx:=cdr(xx);
   eval(v);
   go(exec-loop);
advance:
   xx:=cdr(xx);
   go(exec-loop))

go(x) = setq(xx;cdr(*assoc(x)))
return(x) = prog2(setq(xx;NIL);setq(val;eval(x)))
```

プログ変数の初期化を行ない，次にラベルをさがして各々stackへbindしgo文にそなえ，そののち順に評価を行なうものである。（この引数xを2回走査する方法の他に1回の走査で済ませることも可能である。）
2回の走査を仮定すると，1回目のラベル探索時には実行文を読みとばしており，2回目の逐次実行時にラベルを読みとばすという無駄が目に付く。またgo文の処理は遅くかつ誤りを犯す危険がある。（束縛スタックには変数の属性の区別がないので，未定義ラベルへのgo文があってかつ外側で別に保持されている場合）
中間言語域とラベル連想子を用意し，一回目の走査では各々の登録を行ない，二回目はアレイの各要素の逐次評価をすることにすれば無駄なリスト探索を省き，かつ正しく処理することができる。この結果ALPS/Iでのprogインタプリタの概略は次のようになっている。

```
prog(x) = prog((pform;val;yy);
   yy:=ppoint;
   (car(x)→ progn(ppoint:=addl(ppoint);progtbl(ppoint):=cons(LOCAL;car(x))));
   pform:=x;
   (null(cdr(x))→ return(NIL));
   mapc(cdr(x);λ((j);(atom(j)→ hstore(j;pform;ppoint);
                     T→ progn(ppoint:=addl(ppoint);progtbl(ppoint):=j))));
exec-loop:
   xx:=addl(xx);
   eval(progtbl(xx));
   (eq(xx;ppoint)→ return(val));
   go(exec-loop);
   )

go(x) = (atom(x*)∧v:=hdotp(x*;pform)→ progn(yy:=v;progtbl(xx):=cons(**GO;v));
        T→ ppoint:=hdotp(eval(x);pform))
**go(x) = xx:=x
return(x) = progn(setq(yy;ppoint);setq(val;eval(x)))
```

progtbl はプログ中間言語を表わす。ppointは progtbl 用のポインターとする。x* はxそれ自身をさす。hdotp[x;y] は (x.y)を鍵部にもつH-分子の値部を値とする関数とする。対応する分子がなければ NIL。
hstore[a;0;v] は (a.0)を鍵部に，vを値部にもつH-分子を生成し，値vとする関数とする。

go文はムハ6型であり，アトム引数のgo文に対しては中間言語上で無条件分岐(**GO) への書き換えが行なわれる。この結果ラベルの処理に関しては O(n) から O(1) に高速化され(nはその時までにつまれたスタックの要素数)，実行ループに関してはリストのたぐりがなくなったためメモリ読み出しの回数を半分に減らすことができた。標準問題であるsequenceは プログ形式が主ルーチンであるのでこの改良が大きく影響しており，この問題に関しては同程度のメモリ容量をもつミニコンLispよりハ2倍～ハ8倍速い実行結果が出ている。

⑤即時回収コレクタ：後藤英一教授の示唆(bit誌Lisp入門)に基づき，ALPS/Iには即時回収コレクタが含まれている。このコレクタの目的は次のようなものである。全セルのマーキングによりガベジコレクションを行なう通常のガベジコレクタは所要時間も大きく，その間Lispの処理の進行を止めてしまうので(1)ガベジコレクションの機会を減らすこと (2) ガベジコレクションのアルゴリズムを良くすることが必要である。ガベジコレクションタイムがインタプリタ処理時間を越えては，ちり集めシステムの様を呈してしまう。

このため即時回収コレクタはインタプリタ内で消費するセルを，不要になった時点で即時回収する。evalの中のevlis は実引数のリストを作成するために呼び出されるもので，実行後は不要になる。evlis の定義を次のように変えることによりオーバーヘッドを余り増すことなしにこの操作を行なうことができる。

```
evlis(m) =
   (null(m)→ setq(*last;NIL);
    null(cdr(m))→setq(*last;list(eval(car(m))));
    T→ cons(eval(car(m));evlis(cdr(m))))
```

*lastにはこの結果はevlisの値のリストの最後のドット対が与えられる。この*lastのアドレスのcdr部はかならずNILであり，この部分を現在のFreelistのトップを示すポインタでrplacdすればよい。このポインタをFreeltop，evlis[m]の結果をm'としてこの過程は 次のように書ける。

```
progn(rplacd(*last;freeltop );setq(freeltop;m'))
```

即時回収コレクタにより例えば，sequenceの問題では約20%のセルが回収されている(sequence-20, sequence-40の結果のダンプリストより確認)。sequence-100においてもガベジコレクタの作動は不要であり，物理的には56Kセルであるリスト領域が即時回収コレクタにより意味的に拡げられ，70Kセル程度の大きさを持つとも考えられる。

M式による定義を直線的にコーディングすると，手続きの見やすさが上がる半面，束縛動作の多いプログラムの場合にリストの消費が多くなり，スペース使用効率が悪くなるが，この点をこのコレクタが補っている。(再帰コールは8080のcall命令とpushpopによりそのまま記述できる。)

⑥ハッシュ化配列：ALPS/Iの設計時に次の3つの点があげられ，その解としてHLispを参考にして考え出されたのがハッシュ化配列である。

1) アレイ機構の組み込みの必要性への対応
2) User-propertyの組み込みの必要性とアトム・plistの廃止の対策
3) ユニークオブジェクトの統一的な扱い

ハッシュ化配列の各要素の形式は属性部に「ハッシュ化配列要素」，鍵部に「ハッシュ化配列名」と添字になるS式のドット対をもつことでALPS/I型H-分子に統一されている。

ハッシュ化配列に対しては array, dearray, seta, deleteの4つの関数がある。

array	: fsubr	配列宣言文
dearray	: subr	引数で指定したハッシュ化配列全要素の消去
seta	: fsubr	seta[array-name; subscript; value] 要素への値のセット。array-name以外は評価される。
delete	: fsubr	delete[array-name; subscript] 要素の消去，subscriptは評価される。 array-name[subscript]で占有されていたH-領域中のスロットは自由化される。

箱入娘の解法においては盤面用の配列，差集合をとるためのマーキング，盤面番号の変換表，次に動きうる手を定める手続きの格納などに使われている。

⑦試行用汎関数Try

ALPS/Iは次のような汎関数Tryをもっている。

```
try(x;fn) = prog((xx);
      xx:=x;
loop:(null(xx) → return(NIL);
      fn(car(xx)) → return(car(xx))];
      xx:=cdr(xx);
      go(loop)  )
```

リストxのcarエレメント中で関数fnに対してNIL以外の値を持つものがあればそれを値として返す。なければ"NIL"である。これは集合の中から条件に合うものを1つだけ抜き出すような場合に便利であり，又速い。(この操作はInterLispの catch[mapc[x:λ[[j];[fn[j] → throw[j]]]]] でも可能である。)

⑧連想計算

ALPS/Iでは連想計算を行なう1引数の関数定義体に対してhexprという属性が設けられ，その処理が各々apply,eval中に組み込まれている。hexprとして定義された関数は例えばapplyにおいては次の手順により処理される。(引数を評価することを加えればevalでも同様である。)

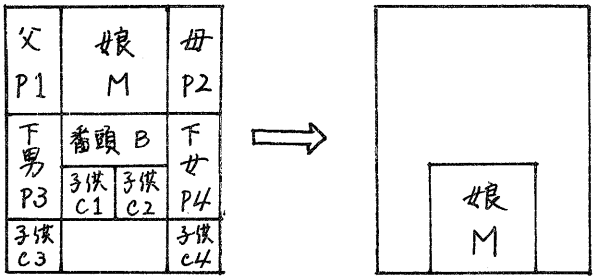
```
eq(get-attrib(fn);HEXPR) →
        (hdotp(fn;car(args)) → hassoc(fn;car(args));
         T → progn(v:=apply(value(fn);args);hstore(fn;car(args);v)))
```

ALPS/IのH-分子はHLisp型H-分子と異なり，ドット対そのものではなく，三つ組であり，Listセルのmonocopyははいっていない。このためリストを引数とする連想計算においては，リストの一意性が保証されていないので注意を要する。

3. 箱入娘パズルについて

「箱入娘パズル程度の問題が解けないようでは，実際に意味のある記号処理の問題を処理するシステムとして全く迫力がない。」という後藤先生の御意見に基づき，性能試験用に箱入娘パズルの解法を試みたものである。

箱入娘パズルは，初期配置C_0から娘を門の所へ連れ出した最終配置C_fに至る最短経路を見い出す問題である。



初期配置C_0　　最終配置C_f
（M以外の物の位置は任意）

初期配置C_0から1手で移りうる配置の集合をS_n，S_nに属する配置から1手で移り得る配置の集合を$smove[S_n]$とする。この時S_{n+1}は次式で与えられる。

$$S_{n+1} = Smove[S_n] - S_n - S_{n-1},$$
$$S_0 = \{C_0\},\ S_{-1} = NIL$$

S_nの中にC_fが含まれたら次式に従って，この目的配置に到達する手順として各々S_nよりd_nを1つだけ取り出す。　$d_0 = C_0$, $d_n = C_f$

$$d_n = try[*smove[d_{n+1}]; \lambda[[j]; member[j; S_n]]]$$

計算機によってこのパズルは既に解かれており，$n = 81$である。Lisp系の言語を用いた解としては，LIPQによる19分31秒というものが76年記号処理委員会報告集に報告されている。また，その手続きの概略はbit誌Lisp入門No. 13 (1974)に完全に記述されており，我々もこの手続きをもとに作成した。

```
nextconf(n;snl;sn) = prog((dn);  print(list(n;length(sn)));
                                 (null(sn) → return(NIL);
                                 seto(dn;choosel(final(sn))) → go(z);
                                 seto(dn;nextconf(addl(n);sn;advance(snl;sn))) → go(z);
                                 T → return(NIL));
                             z   printconf(n;dn);
                                 return(choosel(intersection(snl;smove(list(dn))))))
```

ALPS/Iでの解は`nextconf`，ある1つの配置confから一手で動きうる配置の集合を求める`*smove`，配置のパック・アンパックを行なう`*fn2`，駒位置の連続を調べる`connect`などよりなる。各ルーチンの説明は後述する。

さて，箱入娘パズルによる性能試験の項目としては，次のようなものがあると考えられる。

①すべての手続きをその言語で書き表わすことができる。
②手続き構成を支援するツール，関数が備わっている。
③その手続きを動かすのに充分な記憶容量を持つ。
④実用上充分な高速性がある。

①を満たすことは明らかなので②，③，④が問題になると思われる。ALPS/Iのもつ機能のうちハッシュ化配列，関数Tryなどは②に対するサポートになっている。中間言語方式progインタプリタ，オブジェクトロード機能は④に対する

サポートになっている。しかし記憶装置は64Kワードしかなく（カセットMTはソースファイルに使用），残念ながら8パズル解の初期配置から解を求めることができない。（各配置は1語にパックし，数として記憶するのが一番簡潔かつ高速であるが数の保持は1024しかできない。配置を（M×B，p1×p2，p3×p4，S）の型でリストにして保持するにしても1配置に4語，さらにSnにまとめるために1配置につき1語を要するので d81 を求めるために覚えられる約1万2千の配置を保持するのに60K語リストセルが必要である。）

次に各手続きを説明する。

① *fn2　1つの配置を表わす数 ⇄ M, B, p1～p4, S, x, y　の間の変換を行なう。このルーチンは高速化のためにオブジェクトロード機能を用いて8080機械語により作成されているが，Lispの仕様をまげることはされていない。

例：*fn2[0;dn]：ある配置dnの内容を各変数に分離する

② nextconf　ハッシュ化配列TAGがSn+1を作るためのマーキング（Sn, Sn-1 及びその時点までのSn+1の各配置に対して）に用いられている。不要になったTAGの要素はdelete関数により消去される。H-領域は2Kエントリーを持つので充分に使用できる。また *smove を assoccomp 指定することにより，帰路における再計算を不要にすることができる。

```
FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS...

 DEFINE((
 (NEXTCONF (LAMBDA ( )
 (PROG ( )
     (CSETQ  STACK (CONS SN (QUOTE (NIL))))
     (SETA TAG (CAR SN) 1)
   Y (PRINT N)
     (COND ((NULL SN)(RETURN NIL))
      ((TRY SN(QUOTE (LAMBDA (J)(PROGN (*FN2 O J)(EQ M 13)))))(GO X)))
       (CSETQ VALUE NIL)
     (MAPC SN (QUOTE (LAMBDA (J)(MAPC (*SMOVE  J)
                 (QUOTE (LAMBDA(K)(COND ((NOT (EQ (ERRSET (TAG K) T) 1))
       (PROGN (SETA TAG K 1)(CSETQ VALUE (CONS  K VALUE))))))))))
     (MAPC SN1 (QUOTE (LAMBDA (J)(DELETE TAG J))))
     (CSETQ N (ADD1 N))
     (CSETQ SN1 SN)
     (CSETQ SN VALUE)
     (CSETQ STACK (CONS SN STACK))
     (PRINT SN)
     (GO Y)
   X (CSETQ STACK (CDDR STACK))(PRINT (QUOTE $$$ YATTAZO $))
     (PRINT N)(PRIN1 (QUOTE M=))(PRIN1 M)(PRIN1 (QUOTE B=))(PRIN1 B)
     (PRIN1 (QUOTE P1=))(PRIN1 P1) (PRIN1 (QUOTE P2=))(PRIN1 P2)
     (PRIN1 (QUOTE P3=))(PRIN1 P3)(PRIN1 (QUOTE P4=))(PRIN1 P4)
     (PRIN1 (QUOTE S=))(PRINT S)
     (CSETQ DN (TRY SN1 (QUOTE (LAMBDA (K)(TRY (*SMOVE K)(QUOTE (LAMBDA(J)
          (PROGN (*FN2 O J)(EQ M 13)))))))  ))
   Z (CSETQ N (SUB1 N))(CSETQ SN SN1)(CSETQ SN1 (POPUP NIL))
     (PRINT N)
     (*FN2 O DN)
     (PRIN1 (QUOTE M=))(PRIN1 M)(PRIN1 (QUOTE B=))(PRIN1 B)
     (PRIN1 (QUOTE P1=))(PRIN1 P1)(PRIN1 (QUOTE P2=))(PRIN1 P2)
     (PRIN1 (QUOTE P3=))(PRIN1 P3)(PRIN1 (QUOTE P4=))(PRIN1 P4)
     (PRIN1 (QUOTE S=))(PRINT S)
     (MAPC SN (QUOTE (LAMBDA (J)(DELETE TAG J))))
     (MAPC SN1 (QUOTE (LAMBDA (J)(SETA TAG J 2))))
     (CSETQ DN (TRY (*SMOVE DN)(QUOTE (LAMBDA (J)(EQ (ERRSET (TAG J) T) 2)))))
     (COND ((NULL DN)(RETURN NIL)))
     (GO Z))))))

END OF EVALQUOTE, VALUE IS..
(NEXTCONF)
```

③ *smove　conf から移りうるすべての配置の集合を値とする関数。M,B,P1～P4についての移動可能配置は現在位置をもとにハッシュ化配列を引用し，手続きを取り出し実行させる。これは演算ルーチンの速度に比べ配列参照の方が速いこと，このために必要とされるM,B,Pに対する手続きのセル数は3K程度で済むことなどによる。手数C1～C4の移動に関しては，すきま(x,y)との位置計算により行なう方が速いのでハッシュ化配列は使用しない。

```
FUNCTION EVALQUOTE HAS BEEN ENTERED. ARGUMENTS...
DEFINE((
 (*SMOVE (LAMBDA (CONF)
 (*PROGN
     (CSETQ VAL NIL)
     (*FN2 0 CONF)
     (CSETQ WW ((MM M) NIL))
     (COND (WW (CSETQ VAL (CONS WW VAL))))
     (CSETQ WW ((BB B) NIL))
     (CSETQ VAL (NCONC WW VAL))
          (CSETQ NOWP 1)
     (CSETQ WW ((PP P1) NIL))
     (CSETQ VAL (NCONC WW VAL))
     (CSETQ NOWP 2)
     (CSETQ WW ((PP P2) NIL))
     (CSETQ VAL (NCONC WW VAL))
     (CSETQ NOWP 3)
     (CSETQ  WW ((PP P3) NIL))
     (CSETQ VAL (NCONC WW VAL))
     (CSETQ NOWP 4)
     (CSETQ WW ((PP P4) NIL))
     (CSETQ VAL (NCONC WW VAL))
     (CSETQ NM (OLD M))
     (SETA MARK NM 1)
     (SETA MARK (ADD1 NM) 1)
     (SETA MARK (SUM NM 5) 1)
     (SETA MARK (SUM NM 6) 1)
     (CSETQ NX (OLD X))
     (SETA MARK NX 1)
     (CSETQ NY (OLD Y))
     (SETA MARK NY 1)
     (CSETQ NB (OLDB B))
     (SETA MARK NB 1)
     (SETA MARK (ADD1 NB ) 1)
     (CSETQ OP1 (OLD P1))
     (CSETQ OP2 (OLD P2))
     (CSETQ OP3 (OLD P3))
     (CSETQ OP4 (OLD P4))
     (SETA MARK OP1 1)
     (SETA MARK (SUM OP1 5) 1)
     (SETA MARK OP2 1)
     (SETA MARK (SUM OP2 5) 1)
     (SETA MARK OP3 1)
     (SETA MARK (SUM OP3 5) 1)
     (SETA MARK OP4 1)
     (SETA MARK (SUM OP4 5) 1)
     (CSETQ L 23)
     (CSETQ CLIST NIL)
 FINDC
     (COND ((NOT (OR
        (EQ (MARK L) 1)
        (EQ L 4) (EQ L 9)(EQ L 14)(EQ L 19)      ))
                    (CSETQ CLIST (CONS L CLIST))))
     (SETA MARK L 0)
     (COND        ((ZEROP L)  (GO FINDX)))
     (CSETQ L (SUB1 L))
     (GO FINDC)
```

現在のM,B,Pの位置に対応してハッシュ化配列MM,BB,PPの要素の手続きにより各々WWに移動可能配置をsetしVALへ結合させる。

↓以下の説明：一語にパックされたconfは手数の位置でなくすきまの位置関数Sを保持しているので，各駒とすきまの位置に目印をつけることにより C1～C4 の位置を探し，それをCLISTにまとめる。このために配列MARKが用いられている。MARKには初期値ゼロが実行に先立って与えられている。配列OLD,OLDBはCの位置を求めるための統一化された駒位置の番号への変換を行なう定数テーブルである。

基本位置番号
(左下はB,他は右上)

0/0	1/1	2/2	3/-
4/3	5/4	6/5	7/-
8/6	9/7	10/8	11/-
12/9	13/10	14/11	15/-
16/12	17/13	18/14	19/-

OLD
OLDB

0	1	2	3
5	6	7	8
10	11	12	13
15	16	17	18
20	21	22	23

例 OLD(4)=5
OLDB(9)=15

Cの位置のための盤面位置番号

次ページへ続く

```
FINDX
    (CSETQ PREDXY (CONNECT NX NY))
    (MAPC CLIST
      (QUOTE (LAMBDA (C)
          (PROGN
              (CSETQ OC (NEWC C))
              (CSETQ PREDCX (CONNECT C NX))
              (CSETQ PREDCY (CONNECT C NY))
              (COND ((AND PREDCX PREDXY)
                      (CSETQ VAL (CONS (*FN2 6 OC Y)(CONS (*FN2 6 OC X) VAL))))
                     ( PREDCX (CSETQ VAL (CONS (*FN2 6 OC Y) VAL)))
                     ((AND PREDCY PREDXY)
                      (CSETQ VAL (CONS (*FN2 6 OC X)(CONS (*FN2 6 OC Y) VAL))) )
                     (PREDCY (CSETQ VAL (CONS (*FN2 6 OC X) VAL)))
                  )))))
    (RETURN VAL)
 )))))

END OF EVALQUOTE, VALUE IS..
(*SMOVE)
```

connect[x;y]はxとyが隣接していればT，そうでなければNILを値とする関数である。 connect[x;y] = |x−y|=1 ∨ |x−y|=5

④ハッシュ化配列によるM,B,Pの移動可能配置の並べあげと現在位置による分割。実行に先立ち配列MM, BB, PPの要素をセットしておく。

次に示すのはBBのセットの一部で，例えばBが盤面の左上にいる時(B=0：基本位置0と1を占める場合)に(BB B)の評価により呼び出される値を与えている。もしすきまが下側に2つ並んでいたら(S=71)，そこへ移動させた配置を生成する。もしすきまの一方が右横にある場合(x=2)には一つ右へ移動した配置を生成し，さらに他方のすきまもその右に隣接している場合にはそこまで移動した配置の生成も付け加えられる。

```
FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS..

PROGN (
   (SETA BB 0
        (QUOTE (LAMBDA ( ) (PROGN (CSETQ WK NIL)
         (COND ((EQ S  71) (CSETQ WK (CONS (*FN2 2 3   1) WK)))
               ((EQ X  2) (PROG2
                          (COND ((EQ Y  3) (CSETQ WK (CONS (*FN2 2  2   1
                                                                ) WK))))
                          (CSETQ WK (CONS (*FN2 3  1  0 Y) WK))
                                                                        )))
                 ))      ))
        :
        :  (以下省略)
```

以上のような箱入娘パズルの解法のためにH-分子212個，リストセル5200個が使用される。

⑤実行例　次ページに付録として添付する。

4. まとめ

PROM中に書き込まれたLisp処理系はこれを機会に再点検され，デバックされた。現在の青山学院におけるLispプログラムの処理という観点から見て記述力としては問題ないが，やはり大型の問題を解くには記憶容量を増す方策を立てる必要がある。(速度はミニコン程度という感じである。付録参照)　現在例えば，次の

いくつかの検討をすすめており，別の機会に報告を行ないたい。

①フロッピーディスクを接続し仮想化を行なう。ないしはDISK入出力命令をいれる。

②バルクメモリの配置とデータタイプの分類を変える。フリーストレージ32K，H-領域32Kとし，数をH-分子としてH-領域中にとる。（スタック等はRAMへ持って行くが別のメモリをつける。）

謝辞　日頃御指導頂く経営工学科間野浩太郎教授に感謝いたします。

付録　実行例

実行にはいる前にS_n=1Co，S_{n-1}=NIL，N=0がセットされる。

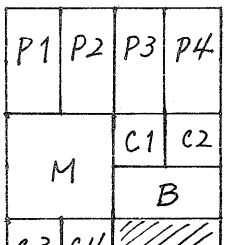
```
FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS...
ASSOCCOMP((*SMOVE))

END OF EVALQUOTE, VALUE IS..
T

FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS...
NEXTCONE NIL
0
(018EB023H 018BBC23H 018BBD23H)
1
(018BBB23H 018BB923H 018BBA23H 018E9523H 018E9623H 018E9E23H 018E9D23H)
2
(018E4127H 018E3A2BH 018E4227H 018E3263H 019E8023H 018E29A3H 018E3163H 018E9223H
 01CB7D23H)
3
(41CB0923H 05CB1A23H 01CB8823H 01CB7E23H 028E17A3H 018E3AA2H 819E0423H 419E0C23H
 019EA623H 019E8423H 018E2E63H 018E293BH 018E3D27H)
4
(028E173BH 018E673BH 018E2D3BH 018E2667H 019E8523H 019EB923H 019EA723H 419E1023H
 809E1723H 018E74A2H 018E3EA2H 128E04A3H 01CB2B63H 01CB7F23H 05CB1C23H 01CB2C63H
 01CB9223H 01CB8923H 45CB0123H 41CB0A23H)
5
(01CB3C27H 01C8B023H 01CB2E63H 01CB3D27H 05CB1463H 05CB1D23H 41CB0B23H 01CB3B27H
 41CB0263H 018E42A2H 018E9EA2H 018E78A2H 809E2913H 019CBD23H 019CBE23H 0190BD23H
 0198A223H 419E1123H 018E313BH 018E953BH 018E6B3BH 128E043BH)
6
(019E803BH 018E9E3AH 4190BE23H 01984627H 0198B823H 0198A123H 0190B623H 0190BB23H
 0190BE23H 019DBB23H 019CB823H 019CB723H 019CB623H 809E3A12H 018E95B2H 41CB0327H
 01CB2667H 01C8B323H 01C8B723H 01C8B823H 01C8B423H 05CB1527H)
7
(01D8A623H 01C8BE23H 019E80B2H 018E29BAH 018E31B6H 809E7412H 809E3E12H 019CA023H
 019CA123H 019C9E23H 019CA223H 019EB923H 019DA923H 019DB523H 0190B823H 0190BC23H
 0190A923H 0190BA23H 0190A023H 0190B523H 0190B723H 0198A023H 0198BE23H 0198B723H
 01984227H 01984527H 4190B823H 4190B723H 4190BC23H 4190BD23H 018E3ABAH 018E427AH
 819E043BH 419E0C3BH 019EA63BH 019E843BH)
8
 YATTAZO
8
M=13B=8P1=0P2=1P3=2P4=3S=166
7
M=12B=8P1=0P2=1P3=2P4=3S=179
6
M=12B=8P1=0P2=1P3=2P4=3S=176
5
M=12B=11P1=0P2=1P3=2P4=3S=146
4
M=12B=11P1=0P2=1P3=2P4=3S=136
3
M=12B=11P1=0P2=1P3=2P4=3S=125
2
M=8B=11P1=0P2=1P3=2P4=3S=185
1
M=8B=11P1=0P2=1P3=2P4=3S=188
0
M=8B=11P1=0P2=1P3=2P4=3S=190

END OF EVALQUOTE, VALUE IS..
NIL
```

Coの実行時間は印刷時間を含めて約3分20秒である。(内，往路3分，帰路は完全に印刷待ちで20秒) 所要メモリはリストセルが44K，H-分子が52Kエントリーである。*Smoveの連想計算を指定しないと帰路が約1分を要し，4分の所要時間，リストセル55K，H-分子334Kエントリーになる。連想計算はL-分子の消費を60～70%におさえている。また，この時得られた配置の総数が133，所要時間3分(往路)より81手解の場合は約300分程度と推測される。



初期配置Co。