会話型LISPの実現と そのGrammatical Inferenceへの応用

太田義勝，中山晶（名大工），吉田雄二（名大計算センター）
福村晃夫（名大工）

§.1 まえがき

会話型のLISP処理系と、プログラム開発とデバッグを支援するLISP向きの強力なエディタとから構成されるLISPシステムを、中型計算機システム（FACOM230-38）の上に実現した[1]（以下、本論文では、このシステムのことをLISP38と記す。）LISP38システムは、会話型処理とエディタにより、プログラムの開発とデバッグを能率的にし、実際的な応用が可能なシステムとなることを目標として構成された。

以下、本論文では、初めにLISP38システムの概要について述べ、次に本システムに用意されたエディタについて述べる。最後に、本システムの実際的な規模の問題への応用の例として、線図形生成文法の推定システムを、本システムに基づいて実現した結果について述べる。

§.2 LISP38処理系の概要

LISP38は、Eval形式の入力を解釈実行するインタープリタを中心に構成されている。処理系の設計に際しては、LISP1.5を基本とし、これにINTERLISP[2]の機能を参考にして、仕様等を決定している。

以下では、LISP38処理系における特徴的な部分を中心に、LISP38の概要を述べる。

§2.1 言語仕様

(1) *implicit-progn*

INTERLISPに用意された機能で、ラムダ形式および条件式の簡単な拡張である。すなわち、関数（LAMBDA arglist* f_1^* f_2^* … f_n^*）が実行されると、形式 f_1^*, f_2^*, … f_n^* が順に評価され、f_n^* の値が関数値となる。また、条件式（COND(P_1^* e_{11}^* e_{12}^* … $e_{1m_1}^*$)(P_2^* e_{21}^* e_{22}^* … $e_{2m_2}^*$) … (P_n^* e_{n1}^* e_{n2}^* … $e_{nm_n}^*$)) は、P_1^*, P_2^*, …, P_n^* を順に評価して、はじめてNILでなくなる P_i^* について、e_{i1}^*, e_{i2}^* … $e_{im_i}^*$ を順に評価して、$e_{im_i}^*$ の値が、条件式の値となる。

(2) ユーザ定義関数

LISP38では、関数の形式として、expr, expr*, fexpr, fexpr* の4種類がある。prefix 'f' は引数を評価しないことを示し、suffix '*' は引数の個数が不定であることを示す。例えば、plusはexpr*を，whileはfexpr*を用いて記述できる。

(3) プログラム制御機能

基本的には、go[l]によるが、ループの構成用に、while[$x_1, x_2, \cdots, x_m$], repeat[$x_1, x_2, \cdots, x_n$]が用意されている。whileの場合、x_1がNILでない限り $x_2, \cdots, x_m$ の評価をくり返す。一方、repeatは、x_n がNILでなくなるまで、$x_1, \cdots, x_{n-1}$ の評価をくり返す。

(4)組み込み関数

現在、50種類以上の組み込み関数がある。表1は、これらの名前のリストである。

表1　組み込み関数のリスト

(subr)	DIFFERENCE	EJECT	AND
CAR	MINUS	REWIND	OR
CDR	QUOTIENT	(subr*)	TRACE
CONS	ADD1	LIST	UNTRACE
ATOM	SUB1	APPEND	WHILE
EQ	ZEROP	NCONC	REPEAT
NULL	GREATERP	PLUS	PROGN
READ	LESSP	TIMES	PROG
PRINT	REMAINDER	(fsubr)	LOADFNS
PRIN1	ABS	QUOTE	SAVEFNS
TERPRI	EQUAL	SETQ	(prog feature)
RPLACA	MEMBER	EDITF	GO
RPLACD	GETD	TIME	RETURN
GENSYM	PUTD	(fsubr*)	(expr)
EVAL	GETP	COND	PRETTY_PRINT
MINUSP	PUTP	DEFINE	FN_CALL_TREE

(5)アトム

文字アトムは、3乙文字以内の英数字名である。一方、数値アトムとしては、−9999以上，+9999以下の整数アトムのみが使える。整数アトムは、ポインタ自身が、整数の値を示している。

§2.2　システムの実現

LISP38システムは、中型計算機FACOM z30-38のOS II/VS上に実現され、会話処理には、システム提供のオンライン機能(SOM)を用いている。システムは、全てFORTRANで書かれていて、全体で約3000ステートメントである。システムの規模は、プログラムエリアが約55kB，データエリアが約105kBである。

図1に、LISP38システムの機器構成を示す。

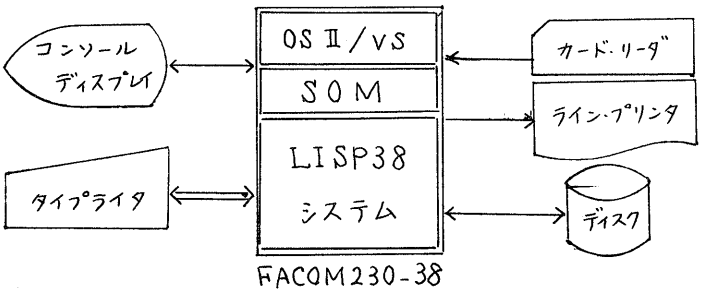


図1　LISP38システムの機器構成

(1) リスト

リストセルは、4バイトで構成され、最大 12 Kセルまで使える。

(2) 文字アトム

文字アトムは、8語より構成され、最大 500 個まで使える。

図2に、その構成を示す。

値セル*	p-list
文字数	ストリングポインタ ← p-name
関数タイプ**	関数定義 ← 定義セル
	(エディタ使用)

図2 文字アトムの構成

* 変数束縛の項を参照。

** ユーザ定義の4つの関数タイプの他に subr, subr*, fsubr, fsubr* の4つの組み込み関数のタイプがある。

(3) 変数の束縛

INTERLISPでは、変数の束縛に、スタックバインディングを用いているが、LISP38では、シャローバインディング(いちばん新しい値を、その変数の値セルにもっていて、束縛の際に、古い値は、スタックの中に退避させる方法。)を用いている。また、シャローバインディングは、prog 中のラベルとステートメントリストとの束縛にも使われている。この方法は、自由変数へのアクセスが速いことに特色がある。

変数の束縛に関連して、LISP38処理系では、セルの消費を、極力抑制するため、引数の評価から、変数との束縛に移る際に、引数リストを作らずに行う。したがって、S-式の入力時、cons, list, append 実行時を除けば、セルが消費されることはない。

§ 2.3 サービス機能

LISP38には、ユーザの便宜のため、次の3つの機能(いずれも、LISP38の関数として定義されている。)が、用意されている。

(1) time[x;n]

time[x;n]は、計算時間と使用したセルの量を測定するLISP38の組み込み関数である。この関数は、式xをn回評価して、GC, I/O時間を除いた平均の計算時間と、平均のセル使用量を印刷し、最後の式xの値を値とする。

また、本論文中の計算時間、セル使用量のデータは、すべてこれを用いて測定を行なった。

(2) pretty-print[fn]

pretty-print[fn]は、ユーザ定義の関数fnの関数定義を pretty-print する関数である。pretty-print は、LISP38の expr で書かれている。

(3) fn_call-tree[fn; yesfns]

fn-call-tree[fn; yesfns] は、ユーザ定義の関数fnの実行に伴って呼びだされる関数の間の関係を樹状に印刷する関数である。引数 yesfns は、追跡の対象とする関数のリストである。fn-call-tree[fn; yesfns]は、LISP38の expr で書かれている。

図3に、その出力例をあげる。

```
FN_CALL_TREE
    FN_CALL
        ALLFNS
    PRINT_TREE
        P_TREE
            P_TREE_SON
                P_TREE
                P_TREE_SON
```

図3　fn_call_tree 出力例

§2.4　計算時間の例

本システムの性能を示す一つの目安として、Sort と Bit_A の計算時間と、セルの使用量を、図4に示す。これら2つのプログラムは、いずれも文献(4)によっている。

計算時間は、ミニコンで実現されているLISPシステムと、ほぼ同じ程度と思われる。またセルの消費量は、Sort のプログラム（Sort_100）が、G.C.なしで計算できるほどになった。（文献(1)のシステム（フリーセル 9.5Kセル）では、2回の G.C. が必要であった。）

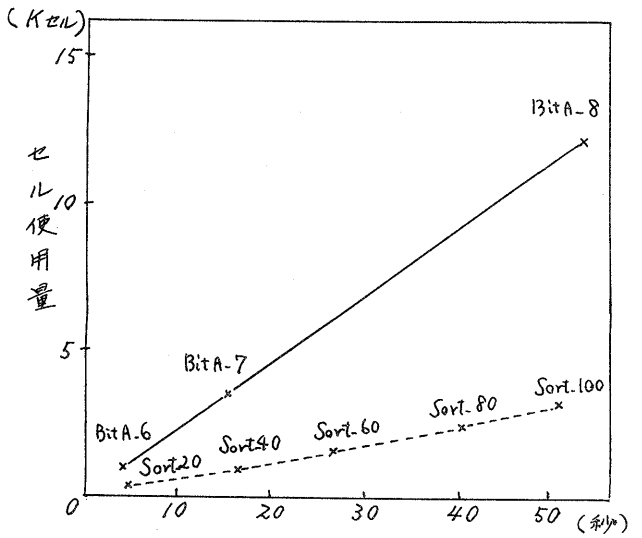


図4　Sort, Bit_A の計算時間,セル使用量

§3　エディタ

会話型のシステムにおいて、エディタは不可欠である。LISP38システムのエディタは、INTERLISPのエディタをモデルとした、リスト編集用の会話型エディタが、LISP38の組み込み関数 editf[fn] として組み込まれている。このエディタは、デバッグの際に関数定義の編集等に使用することができる。なお、このエディタは、現在、FORTRAN で書かれている。

エディタは、編集しようとしている関数定義のリスト中のある部分リストを指すポインタを持っていて、すべてのリスト編集コマンドは、この部分リストに対して、適用される。

エディタコマンドとしては、印刷、ポインタの移動、リストの挿入、除去、置き換え、カッコの編集, UNDO など、約30種類ある。

これらのうちで、特に特徴的な、パターンマッチングによるポインタの移動と、UNDOコマンドについて、以下に簡単に記す。

§3.1 パターンマッチングによるポインタの移動

編集を行なうためには、ポインタが、目的の部分リストを指すようにしなければならない。このための基本的な機能を与えるものとして、1レベルずつ、ポインタを移動させるコマンドがあるが、より実用的な機能として、目的のリストとマッチするパターンを与えて、そのパターンとマッチするリストの位置までポインタを移動させるコマンドがある。

パターンは、記号"&"と"--"を含む任意のリストで、"&"と"--"は、それぞれ任意の要素、および任意のリストセグメントとマッチする。例えば、(CAR &)は、(CAR(CDR(CDR X)))とマッチし、(COND --)は、(COND((ATOM X)X)(T(FF(CAR X))))とマッチする。

§3.2 UNDOコマンド

リストの挿入などのコマンドは、編集前のリスト構造を破壊してしまうのでコマンドの入力を誤った時、そのコマンドの効果を元にもどすことが必要である。そのためのコマンドとして、UNDOコマンドがある。このコマンドは、編集対象のリストを、最も最近におこなわれた破壊を伴うコマンドの実行前の構造に復元する。破壊を伴うコマンドとしては、リストの挿入，除去，置き換え、カッコの編集等があげられる。

§4 Grammatical Inference システム

LISP38システムの有効性を確かめ、かつ実際に具体的な問題を解くために、線図形サンプルの集合を与えられて、それを言語として生成する文法を、自動的に推定するシステムを構成する。なお、線図形生成文法の推定問題については、文献(3)に、既に発表済みであるので、ここでは、このシステムを、特にLISP38側から見た時の結果について述べる。

§4.1 文法自動推定システムの概略

線図形は、8方向指数(図5)により、記号系列表現されていて、さらに、この系列を相続く記号の組を単位としてまとめた系列が、システムへの入力サンプルストリングとなる。(図6)

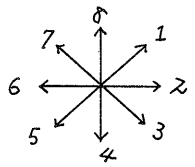


図5 方向指数

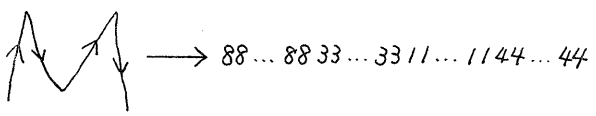


図6 入力サンプルストリング

1つのカテゴリに属する線図形を生成する文法の自動推定は、大きく分けて次の3つのステップにより行なわれる。

(1)サンプルの構造化

サンプルストリングを、記号間の相関に基づいて、グループ分けし、記号をグループ名記号で置きかえる。また、同時に、相関の強さに基づいて、カッコづけを行なう。

(2)サンプル文法の構成

カッコづけられたサンプル1つより、それを導くような文法（サンプル文法）を構成する。

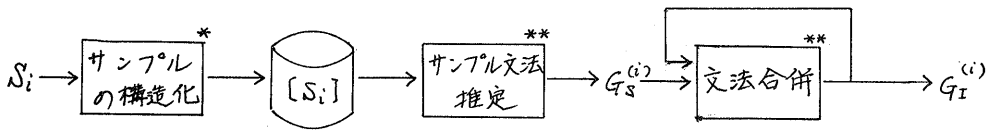
(3)推定文法の構成

あるカテゴリに属するサンプルのサンプル文法を逐次的に併合を行なうことにより、サンプル集合に対する推定文法を構成する。

図7は、このシステムの概略の構成を示している。

上の3つのステップのうち、ステップ(1)は、FORTRANで書かれていて、出力であるカッコづけられたサンプルは、前もって作りあげて、ディスクに書き込まれている。

ステップ(2),(3)は、本システムのLISPで書かれていて、ディスク上のカッコづけられたサンプルを読みながら文法推定を行ない、結果をLPに出力する。



S_i ：原サンプル　　* FORTRAN プログラム
$[S_i]$ ：構造化サンプル　　** LISP プログラム
$G_S^{(i)}$ ：サンプル文法
$G_I^{(i)}$ ：推定文法

図7　文法自動推定システムの概略

なお、本実験に使われたデータは、手書き大文字（1サンプル＝20×30ビット）のうち、1ストロークで標本化された文字14種（C, D, G, I, J, L, M, N, O, S, U, V, W, Z）である。

§4.2　文法推定プログラムの構成

文法推定は、関数 ginfer[category; k] により実行される。ここにcategoryは、文法を推定するカテゴリ（アルファベット名），kは推定に使うパラメータである。

プログラムを構成する関数は、ginferを含めて25個（組み込み関数は除く）で、全部あわせたプログラムのサイズは、リストセルにして、約2kセルである。

また、これらの関数中で用いられる大局的な変数として、次のものがある。

(i)サンプル文法のS.N.表現
(ii)サンプル文法のC.N.表現
(iii)推定文法のS.N.表現
(iv)推定文法のC.N.表現
(v)初期記号の右辺

（S.N. ; Complex Nonterminal
C.N. ; Simple Nonterminal）

以下に、それらの関数の機能と、呼び出し関係を示す。（*印のついた関数は、再帰関数）

ginfer[category; k]
　　データの category，パラメータ k を与えて、データを読みながら、逐次、文法推定を行う。

dsearch[category]	データファイルの初期化
getsample[]	データを1個読む
printn[sn; cn]	文法の印刷
gmergec[], gmerges[]	C.N., S.N. の集合の union をとる。
*smerge[x; y]	ストリングの集合の union をとる。
smember[x; y]	ストリング集合の membership
*seq[x; y]	ストリングの比較
sginfer[sample]	サンプル文法の構成
*kpbs[x]	構造木から k-projection と bracketed string の対を求める。
kpbsjoin[lkpbs; rkpbs]	k-projection, bracketed string の合成
addinf[x; y]	infix の追加
addpref[x; y]	prefix の追加
addsuf[x; y]	suffix の追加
addcn[x; y]	C.N. の追加
sadd[x; y]	ストリングをストリングの集合に追加
core[x]	
prefix[x]	
suffix[x]	
*tails[x]	ストリングの末尾の記号
heads[x]	ストリングの先頭の記号
sconc[x; y]	ストリングの連接
*gconc[x; y]	一般的なストリングの連接
*leftss[x]	右端の要素を除いたストリング

§4.3　文法推定にかかる計算時間、セル使用量について

　パラメータ k を2~4にかえて、文法推定を行なった時の計算時間、セル使用量を表2に示す。

　サンプル(M)について、パラメータの値を2~4にかえた時、各関数の呼び出される回数を表3に示す。

§5　あとがき

　中型計算機上に、FORTRANで書かれたLISPシステムで、実際的な規模の問題が扱えたということは、一応の成果であった。現在、インタープリタのみのシステムであるので、実行時の効率を上げるのに、コンパイラの実現を考慮中である。

§6　謝辞

　日頃、御指導いただく本学本多波雄教授並びに福村・本多両研究室の皆様に感謝します。

表2　計算時間，セル使用量（1サンプルあたり）

サンプルカテゴリ	サンプル数	サンプル平均サイズ	計算時間(秒) k=2	k=3	k=4	セル使用量（セル） k=2	k=3	k=4
C	30	46	2.9	2.9	2.9	434	442	450
D	27	65	4.2	4.2	4.2	606	618	630
G	17	59	4.8	4.6	4.2	572	584	583
I	29	26	1.5	1.4	1.4	239	236	239
J	27	35	2.3	2.2	2.0	341	343	336
L	30	38	2.1	2.1	2.1	340	346	352
M	24	56	3.9	3.5	3.3	547	540	542
N	21	62	3.3	3.3	3.3	537	546	555
O	30	62	4.2	4.2	4.2	589	601	610
S	25	50	4.1	3.7	3.3	498	500	491
U	30	56	3.1	3.2	3.1	500	509	512
V	27	43	2.3	2.3	2.2	377	383	389
W	7	58	3.4	3.4	3.3	527	538	547
Z	26	55	2.9	2.9	2.9	467	476	485

表3　関数の呼び出し回数（カテゴリ M）

関数名	関数サイズ	k=2	k=3	k=4
ginfer	168	1	1	1
dsearch	57	1	1	1
getsample	23	24	24	24
sginfer	74	24	24	24
printn	58	48	48	48
gmergec	118	24	24	24
gmerges	188	24	24	24
kpbs	39	1388	1388	1388
core	8	2370	2116	1994
prefix	6	1404	1392	1390
suffix	10	598	464	384
sadd	18	128	90	80
addpref	101	72	58	52
addsuf	109	72	58	52
heads	21	1094	848	700
tails	21	697	544	477
smerge	42	714	686	633
smember	50	605	586	541
kpwsjoin	448	682	682	682
seq	42	2597	1700	1591
sconc	285	714	690	686
leftss	24	542	422	348
addcn	89	48	34	28
addinf	105	32	8	4
gconc	29	445	293	242

§7　参考文献

(1) 太田，吉田，福村，「INTERLISPの機能をとり入れた会話型LISPの実現」，昭和52年度通信学会全国大会論文集（1317），1977

(2) W. Teitleman，「INTERLISP reference manual」，Xerox Palo Alto Research Center，1974

(3) 中山，吉田，福村，「不規則成分を伴う線図形の文法推定による構造抽出と記述の一方法」，通信学会 パターン認識と学習研究会 PRL 77-29，1977

(4) 中西正和，「LISPコンテスト」，Bit Vol-7，No-3，1975

会話例

```
<-(DEFINE(FACT(LAMBDA(N)(COND((ZEROP N)1)
...(T(TIMES M(FACT(SUB1 N)
(FACT)
<-(FACT 6)
U.B.A.
M
<-(EDITF FACT)
EDIT
*(F(TIMES --)
*P
(TIMES M (FACT &))
*(-2 N)
*P
(TIMES N M (FACT &))
*UNDO
(-2) UNDONE
*(2 N)
*P
(TIMES N (FACT &))
*OK
FACT
<-(FACT 6)
720
<-
```

} 関数 FACT の定義

} エラー・メッセージ，変数 M の値が定義されていない

← エディタの呼び出し

← ポインタを（TIMES M（FACT（SUB1 N））に移動させる

← 2番目の要素の前にNを挿入（誤ちがったコマンド）

← (-2 N)の効果を元に戻す

← 2番目の要素をNで置き換える

← エディタより抜け出す