資　料

LISP 1.9 プログラミングシステム*

黒　川　利　明**

Abstract

LISP 1.9 is an extended and revised programming system from LISP 1.6. Its fundamental features are as follows: 1) high performance both on execution and debugging, 2) many useful functions, 3) new powerful input/output facilities.

LISP 1.9 has also many useful libraries written by users, among which are as follows: 1) graphic input/output facilities, 2) translation facility for programs written in other LISP languages, 3) programming languages for AI application, 4) natural language processing system, 5) mini-question-answering system with figure-output.

The current status of LISP 1.9 system is described in this paper with a discussion about the future extensions. LISP 1.9 is originally implemated for the Pattern Information Processing System Project supervised by Electrotechnical Laboratory.

1. まえがき

LISP 1.9 プログラミング・システムは，基本的には LISP 1.6[4),5),21)] の延長線上にあり，通産省のパターン大型プロジェクトの一環として開発されたものである．LISP 1.9 は大規模実験的システムを作成するためのプログラミング・システムとして開発されたのであるが，本論文では，その基本的機能と，その上に現実にどのような実験システムや，システム作成用の道具が作られているかについて述べる．なおLISP 1.9 の機能でも LISP 1.6 以来の（データ構造とか基本的組込関数とか）機能については，他論文[4),6)] に記述があるので，説明を省略した．

ある程度の知識を用いるような高級なパターン情報処理を行おうとすれば，データベース（貯えられた知識）の管理や記号処理が必要となる．LISP はこのような目的に合致する言語（道具）として，米国では多用されているが，日本においては二，三の例（東京大学の HLISP-REDUCE[2)]，京都大学の PLATON[12)]，電子技術総合研究所[1)] など）を除いては，まだ充分に用いられていない．このなかで LISP 1.9 は，広範囲の応用を前提とし，TSS-インタラクティブな使用を主としている点に特長があり，米国での LISP システムに優るとも劣らぬ機能をもっている．

LISP 1.9 の基本的機能としては，

1) 実行時の高速処理．
2) デバッグのための豊富な情報と有用な関数．
3) 有用な組込関数．
4) 新しい強力な入出力機能．

が挙げられ，LISP 1.9 上の主な応用プログラムとして，ここに紹介させていただくものには，次のようなものがある．

1) グラフィック入出力関数群．
2) 他 LISP 言語により書かれたプログラムを LISP 1.9 に翻訳するシステム．
3) AI 用のプログラミング・システム．
4) 自然言語処理システム．
5) 質問回答システムでデータ・ベースの現状を絵によって示すもの．

LISP 1.9 のこれからの拡張については，

1) 入出力機能の追加，拡充(漢字出力等)．
2) ストリング処理機能の拡充．
3) 処理速度の向上（主としてコンパイラ）．

* LISP 1.9 programming System by Toshiaki KUROKAWA (Information Systems Laboratory, TOSHIBA Research and Development Center, Tokyo Shibaura Electric Co., Ltd.)

** 東京芝浦電気(株)総合研究所情報システム研究所

Table 1 Interpreter Execution Time Comparison on a Sorting Program. (LISP 1.6/LISP 1.9)

ソートする個数	LISP 1.9				LISP 1.6				L1.6/L1.9	
	実行時間 (ms)	GC の回数	GC 時間 (ms)	純実行時間 (ms)	実行時間 (ms)	GC の回数	GC 時間 (ms)	純実行時間 (ms)	GC 時間も含んだ比較	純実行時間の比較
20	409	0	0	409	1,466	1	377	1,089	3.58	2.66
40	1,506	0	0	1,506	7,632	8	2,520	5,112	5.07	3.40
60	2,505	0	0	2,505	12,802	12	4,278	8,524	5.11	3.40
80	3,953	1	124	3,829	20,343	19	7,293	13,050	5.15	3.41
100	5,019	1	134	4,885	26,548	35	9,943	16,605	5.29	3 40

(注1) このソート・プログラムは，情報処理学会の記号処理シンポジウム[5]で行われた LISP コンテストのテスト・プログラムである．
(注2) フリー・リストは，いずれも 4k セルである．

4) トレース機能の強化．

などが考えられている．

2. LISP 1.9 の基本的機能

(1) 実行時の高速処理

システムの基本的機能の一つとして，処理の高速性をとりあげるのは，あるいは奇異に思われるかもしれない．ここで高速性をとりあげる理由には二つあって，一つには LISP 1.9 のような TSS-インタラクティブ・システムにおいては，処理の高速性が後述するようにシステムの使い易さという機能的側面をもつことであり，いま一つには，これまで定説の如くになっている LISP の低速性ということは決して必然的なものではなく，インプリメンテーションによってはかなりの高速性を期待できることを示すためである．

TSS-インタラクティブ・システムにおいて，高速性がいかに重要な因子であるかは，「応答が遅いときには，使う気になれない．」というユーザの声一つをとってみてもわかるであろう．何秒という時間でシステムから応答があって始めてシステムと人間とのダイナミックなインタラクションが可能になる．(バッチ・システムの欠陥の一つはこのインタラクションが何時間とか何日とかいうオーダでなされることである．)

LISP 1.9 の高速性は例えば **Table 1** の LISP 1.6 との比較を見れば明らかである．この高速化は Table 1 からもわかるように次の四点について行われた．

1) インタープリタの高速化．

Table 1 の純実行時間の比較を見ると約 3.4 倍になっている．大きなプログラムでは 5 倍以上になる．

2) ガーベッジ・コレクションの回数の減少．

全実行時間での比較が，Table 1 のような小プログラム (フリーリスト=4k セル) ですら 5 倍以上の処理速度になっている．大プログラムでは 10 倍以上になる．

3) ガーベッジ・コレクションの高速化．

マーキング・アルゴリズムの高速化[7] とフリーリスト作成時の高速化などにより，3 倍程度になった．

4) 基本的操作の高速化．

コーディング上の技術に過ぎないが，TOSBAC-5600 の割り込み機能を用いて，スタック操作や CONS 操作の速度を 2 倍程度にひきあげた．

なお **Table 2** には，MIT-MACLISP, Xerox-INTERLISP という米国での代表的 LISP システムとの比較をのせている．計算機も OS も異なるので正確な評価とはいえないが，一つの目安となるであろう．

インタープリタが，どのように変更されたかの詳細は別論文[8] に譲るが，処理高速化に関する本質的な部分は，不必要なスタック操作，evlis 操作の除去とユーザ定義関数やラムダ表現の処理を eval の内だけで済ませられるようにして，eval-apply の不必要な再帰呼出しを省略したことにある．その他にも，後述のデバッグに関連したブロック情報をスタックに積んだり，いくつかの新しい関数型 (NEXPR* など) を導入し，ユーザの便宜をはかっている．

Table 2 Performance comparison with other LISP systems. (20 items sorting program)

	INTERLISP	MACLISP	LISP 1 9
LISP システム名 (20ケのソーティング)	With spaghetti	PDP-10 Multics	TOSBAC-5600/170
純実行時間 (msec)	2,510	1,563 1,318	409

(注) INTERLISP のデータは，1976 年 4 月 6 日付の W. Teitelman の手紙による．
計算機は，TOSBAC-5600/170 より 2~2.5 倍遅い．
spaghetti-処理を伴わなければ，1 割位速くなるであろう．
MACLISP のデータは，1974 年 11 月 6 日付の D. A. Moon の手紙による．
PDP-10は，サイクル 1.8μsec, 1.4 MIPS, Multico はサイクル 0.5 μsec, 0.65 MIPS で，TOSBAC-5600/170 と同性能．

* NEXPR は EXPR と同様に複数個の仮引数を用い実引数を対応させるのだが，この時実引数の評価 (eval) を行わない．

```
SYSTEM 7L1.9

Version 4.57 (1976/01/14) See lisp/1.9/info.

ready

OLD OR NEW-NEW
READY
standard ? YES
$RUN
? (DE FACT (N) (COND ((ZEROP N) NIL) (T (TIMES N (FACT (SUB1 N]
FACT
? (FACT 3]

240 NON-NUMERIC ARGUMENT FOR ARITHMETIC FUNCTION. TRY (SOSTK 5) !

ready

/ (BTV]
 #066125  break-by-error
0 block #066117 (TIMES N (FACT (SUB1 N)))
1 block #066112 (COND ((ZEROP N) NIL) (T (TIMES N (FACT (SUB1 N)))))
 #066105 N == 2
2 block #066103 (FACT (SUB1 N))
3 block #066076 (TIMES N (FACT (SUB1 N)))
4 block #066071 (COND ((ZEROP N) NIL) (T (TIMES N (FACT (SUB1 N)))))
 #066064 N == 3
5 block #066062 (FACT (SUB1 N))
6 block #066055 (TIMES N (FACT (SUB1 N)))
7 block #066050 (COND ((ZEROP N) NIL) (T (TIMES N (FACT (SUB1 N)))))
 #066043 N ==  #766666
8 block #066041 (FACT 3.)
end-of-stack encounter-#118.
/ N
1
/ (A DEL
(FACT (SUB1 N]
NIL
/ (EXIT 1]
6
? DONE
$DONE
SYSTEM ?
```

Fig. 1 A Usage Example of Break-package

(2) デバッグのための豊富な情報と有用な道具

デバッグというものは決して余分なプロセスと見られてはなるまい．**普通のユーザ**を前提とした TSS システムでは**デバッグ**もプログラム作成時の重要で欠かすことのできないステップである．デバッグには虫の本質や由来を究めることと虫の除去という二側面がある．LISP 1.9 はこの両面を考慮して作られている．

(i) ブレイク・パッケージ

Fig. 1 の例を見るとわかるように LISP 1.9 はエラー処理のための特別な管理ルーチンを用意していて，これはブレイク・パッケージ (break-package) と呼ばれている*．ブレイク・パッケージに入ると通常の入力要請記号(?)とは別の記号 (/) が入力要請として出力される**．ここでは通常の実行時におけると同様に read-eval-print が行われるのだが，このパッケージ中で主として用いられる関数として後に説明する BTB, BTV, SOSTK, WHERE, REGS, EXIT, RTEVAL, CONT, CALLSS が用意されている．

UCFLAG という変数の値を変更すればエラー時にブレイク・パッケージに入らずユーザの用意した ERRSET ルーチンに入ることもできる．

(ii) エラー時の情報収集

どこでエラーが生じたかは虫取りの最も重要な情報である．従来は再度同じプログラムをトレースをかけながら走らせることでこの情報を得ていた．LISP 1.9 ではバックトレースと呼ばれる機能***があってブレイク・パッケージから，これまでにどのような関数形が評価されたかを歴史の新しいものから (時間的に逆転して)出力でき****，トレースをかけなくても済む場合が多い．BTB (Back Trace of Blocks) や BTV がその関数であり，BTV には Fig. 1 のようにブレイク・パッケージに入った原因や束縛変数の値*****，CATCH や ERRSET に入ったかの情報をも出力する．

各ブロック情報はインタープリタが再帰的処理のた

---

* ブレイク・パッケージにはエラー時以外にもブレイク・キーを押した時や BREAK という関数を評価したときに入る．

** これはシステムが標準的にそうしているだけであって後述の PROMPTCHAR という関数によりユーザが随時変更することも可能である．

*** この呼び名は INTERLISP[19] に倣っている．

**** スタックの上から出力するからこうなる．なお Fig. 1 のように上から 0,1,2,… とブロック番号を付けてゆく．

***** LISP 1.9 は shallow-binding を用いているので一つ以前の古い値がスタックされる．なお Fig. 1 の #766666 は unbound-value を示すマークである．

めに戻り先番地をスタックするときに，評価している関数形とブロック・マークとを同時にスタックすることにより貯えられている．

Fig. 1 の例ではどこで非数値引数が発生したかは，このバックトレーシングと変数や関数の現在の値を調べるだけで明白になっている．BTB, BTV の他には SOSTK というスタック上の全情報を出力する関数やエラー発生時のレジスタの内容を出力する REGS という関数や，(BTB 1 0) に等しい WHERE という関数がある．

(iii) エラーの処理

エラーがどのようにして生じたかを知った次の段階はその処理である．一つにはエラーの原因をなくすことであり，いま一つには（可能なら）実行を継続させることである．

エラーの原因をなくすにはエディタが必要である．LISP 1.9 にはリストに対する編集システムはまだ備えられていないが文字列に対する編集は可能で，例えば (CALLSS 'EDIT) というふうに呼び出せる．CALLSS は他にもファイル管理ルーチン，ファイルダンプルーチン等を呼び出せ，さらにはシステムの終了まで要求することができる．

エラー出現後，実行を継続するには 3 種類の方法がある．一つは Fig. 1 にもある EXIT で，これはブレイク・パッケージから，ある値をもって抜けることを意味している．この Fig. 1 の場合にはエラーを生じた 0 ブロックの (TIMES N (FACT (SUB1 N))) の値として 1 を与えてブレイクから抜け出すことを意味していて，これにより実行はあたかもエラーがなかったかの如く終了する．EXIT は第 2 引数として抜け出すべきブロックを指定できる*．一方 RETEVAL は，まず抜け出すべきブロックに束縛変数の値などを回復しつつ戻ってその環境で第 1 引数を評価しその値をもって抜け出す．いま一つの関数 CONT は例えばスタックオーバーフロの割り込みによりブレイクしたとき**，さらに実行を継続するときに用いられる．

ブレイクからの脱出に ERR 等の一般的関数を用いることもできる．**Fig. 2** にはブレイク・パッケージを含む LISP 1.9 のコントロール構造が示されている．

ブレイク・パッケージを実行とは全く別のフェーズにすることも考えられたが，不必要なメモリの消費や煩雑なオーバーレイを避けるためにインタープリタの中に埋め込まれた形式になった．

ブレイク・パッケージに対するユーザの一般的評価は，ある程度理解するまでは，とっつきにくいが，使いこなせるようになればデバッグが非常に速くなるという点で一致している．リストに対する編集機能をも備えればブレイク・パッケージの有用性は一段と高まるであろう．

なおブレイク・パッケージに代表される LISP 1.9 のインタラクティブ・デバッグ機能は，単にデバッグのみならず，既存のプログラムを理解するにも，非常に役立っている．

(3) 有用な組込関数

LISP 1.6 よりも約 30 の新しい関数が追加されている．この中には前述のブレイク・パッケージの関数や後述の入出力関数の他に以下のような関数が，ユーザの要請等により含まれている．

1) プログラムをより柔軟にコントロールできる，CATCH, THRON, PROGN.
2) TOSBAC-5600 のアセンブラに相当する LAP-コードを直接実行する ASSEMBLE.
3) アイデンティファイヤの全属性を出力する ATOMPRINT.
4) アドレスを直接扱える #OUT, #SUB, #SETA 等.
5) チェックをしないで実行する速い版の関数としての FCAR, FCDR.
6) OBLIST について調べる OBCHECK, 直接出力する PRINTT などユーザの要求により作られた関数群.

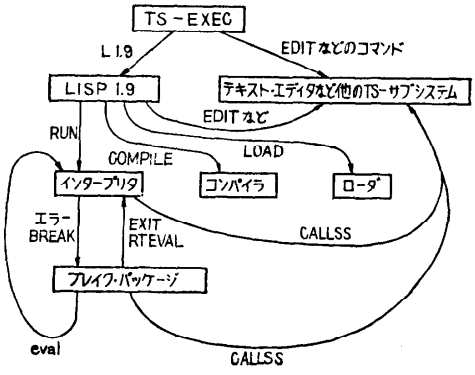

Fig. 2 Control structure of LISP 1, 9

* 指定がないと Fig. 1 のように 0 ブロックとみなす．

** LISP 1.9 ではスタックのサイズは最初最小(1k ワード以下) にとってあって，実行中必要になった時点でスタックのサイズを 1k ごとに増やしていく．STKFLAG というシステム変数の値が 0 なら割り込みは生じないが，1 だと上記の割り込みが生じる．

これらの詳細はマニュアル[22]に記述がある.

(4) 新しい強力な入出力機能

入出力機能はどのような言語においてもインプリメントされる計算機に存在するために,往々にして充分な機能定義がなされないままとなる.しかも入出力機能が充分でないとシステム自体の性能がどんなに良くともユーザにとっては使い物にならなくなるという厄介なものである.

LISP 1.9 では入出力機能を次のように根本的に改良して速度の増大と機能の強化をはかった.

i) 論理チャネルの導入と入力/出力チャネルの分離

これまで入出力はファイルとか端末とか物理的対象ごとに,しかも入出力は同じ対象に行うと考えていた.しかし,これでは例えば端末から入力してファイルに出力する場合には不便であり,さらに同じ端末入力とはいってもインタープリタのトップレベルで読む場合とブレイク・パッケージに入っての入力,ユーザの READ 関数によって入力する場合とでは意味が異なり,相互に無関係であることが望ましい*.

そこで入力/出力チャネルを分離し,さらに論理チャネルという概念を導入して一つの物理チャネルに複数個の論理チャネルを設けることができるようにした.チャネルの宣言は DECCNL という関数で,入力/出力チャネルの設定は INCNL/OUTCNL によって行う.

ii) カレント・チャネルの導入

これまで入出力時には何に入出力するのかを常に与えてきた.これでは read/write のたびに入出力の対手をセットせねばならなかった.これに対して LISP 1.9 ではチャネルの指定が無い場合には現在のチャネルをそのまま使用することにし,これによって処理速度の増大を実現し,かつ最初にチャネルを変更することによって同一のプログラムが端末から入力することもできれば,ファイルから入力することもできるという可変性を付加した(**Fig. 3**).

iii) チャネル属性の強化

これまで入出力での属性的要素としては例えばファイルのラインナンバーを入力時に無視するかどうか,とか入出力時の1行の文字数をどうするかなどがあったが,LISP 1.9 ではこれらの他に以下の属性を付け加え,かつ各論理チャネルごとに独立に与えられるようにした.

- スタート・コラムとエンド・コラムのセット (SET-LINE)
  LINELENGTH の拡張で,何コラムから何コラムまでを入出力の対象にするかを指定する.
- ファイル入出力の監視 (VERIFY-FILE)
  ファイルへの入出力を同時に端末に出力する.
- エンド・オブ・ファイル割り込み (ON-EOF)
  ファイル入力時,エンド・オブ・ファイルになったら指定した割り込みルーチンを実行する.
- 入力要請記号のセット (PROMPTCHAR)
  端末入力時,システムは入力要請記号を出力して,ユーザに知らせるが,この記号をセットできる.
- 端末入力のファイルへの記憶 (MEMO-FILE)
  端末入力を同時にファイルに出力して入力情報を保存しておくことができる.

これらのチャネル属性は PRINTCNL という関数によって調べることができる.他に次のような関数が用意されている.

FIND-EOF——eof をスキップする.

PUT-EOF——eof を書く.(ファイルのクローズに相当する.)

REWIND——ファイルを巻戻す.

3. LISP 1.9 上の主な応用プログラム

LISP 1.9 は,より有効な研究用の道具となるシステムを作るための基本的な道具としてこそ大きな価値がある.したがって LISP 1.9 の価値は「これによってどのように有力な道具やシステムが作成されているか」にあるといっても過言ではない.以下に紹介させ

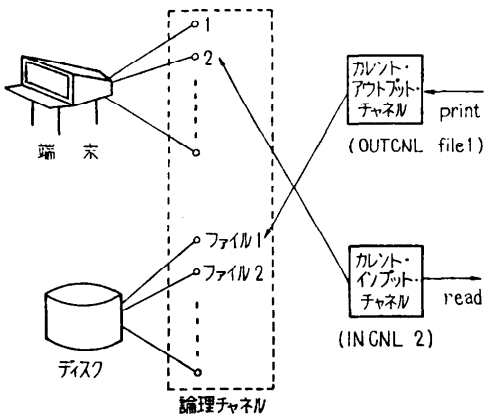

Fig. 3 Channel structure of LISP 1.9

* 従来の方式では,例えばユーザの READ による入力の途中でブレイクして,いろいろ調べようとするとユーザの READ のための入力が,ブレイクのための入力と取り違えられるなどという不便があった.

Table 3 Graphic Input/output Facilies (PLOTL 9)

関数名	引数	値	
ERASE		T	画面消去
INIT		T	イニシャルセット(プログラム実行前に必要)
MOVEA	X Y	X Y	ポイントの移動
MOVER	X' Y'	X Y	〃
DRAWA	X Y	X Y	〃 (線を引く)
DRAWR	X' Y'	X Y	〃 (線を引く)
DRAWT	X Y	X Y	Write-thruモードにて線を引く
LETTER	X Y Σ	X Y	文字(Σ)のライト
GIN		X Y Σ'	カーソル位置の入力
DRAWD	X Y	X Y	ポイントの移動(線を引く) ---------- (4014のみ)
DRAWDD	X Y	X Y	〃 〃 —·—·— 〃
DRAWSD	X Y	X Y	〃 〃 – – – – 〃
DRAWLD	X Y	X Y	〃 〃 ———— 〃
DRAWF	X Y	X Y	〃 〃 ———— 〃
DRAWFD	X Y	X Y	〃 〃 ·····〃
DRAWFDD	X Y	X Y	〃 〃 —·—·— 〃
DRAWFSD	X Y	X Y	〃 〃 – – – – 〃
DRAWFLD	X Y	X Y	〃 〃 ———— 〃
CURSOR		flag X Y	アルファ・カーソルの位置 (flagは画面の左か中央かを示す)
BELL			ベルを鳴らす。

(注) パラメータの説明
X：X位置 Y：Y位置 Σ：文字 (アトム, リスト)
X'：X位置(相対) Y'：Y位置(相対) Σ'：1文字

ていただくシステムは，この意味で LISP 1.9 の価値を作り上げている．もちろん，これらが LISP 1.9 上の応用プログラムのすべてではない．特筆すべきことはこれらが LISP ユーザ (LISP 作成者ではない！) によって作成供給されていることであり，そしてこれらのプログラム作成上での様々な要求やコメントが LISP 1.9 の今日を作りあげていることである．

(1) グラフィック入出力機能 ((株)エス・ジーの三幣恭敏氏作成)

図形端末用ユーティリティ群で，FORTRAN 用の PLOT-10 に相当する．**Table 3** に関数のリストがあるが，このシステムを稼動させるために使用可能な文字の種類を増やしたり，PROMPTCHAR を導入したりした．

(2) LISP 翻訳システム (東京大学の後藤典孝氏 (現 富士通) 作成)

LISP における方言差は割合に大きいが，これを補う方法としてはプログラムを変換する方法がある．プログラム変換システム自体，LISP で書けるので他のプログラム言語の場合に比べて容易に作成できる．このシステムは MACLISP[10] で書かれたプログラムを LISP 1.9 に翻訳するもので定義した関数のリストの出力や文法チェック，若干のオプティマイズをも行う．

(3) AI 用のプログラミング・システム (電子技術総合研究所の島田俊夫氏がインプリメント))

Micro-Planner[15),23)] と Conniver[9)] がある．

PRINTT は Micro-Planner の，CATCH, THROW は Conniver のために組込まれた．

(4) LINGOL[3),13),14)]

もとは MIT で開発された自然言語を処理するシステムであるが，電子技術総合研究所の田中穂積，淵一博氏等によってインプリメントされ，日本語の文章解析のために用いられている[1),16)]．また LINGOL を用いて Winograd の積み木の世界[20)]に対する入力文を解析する SHDRLV というシステムも作られている．

このシステムはストリング処理，アレイ処理を多用しており，OBCHECK はこのために作成された．

(5) MILISYJS[1),17),18),24)]

これは MILISY[11)] を基礎にして日本語による質問回答システムをつくりあげたもので，データ・ベースの図示機能を備えている．このプログラムは LISP で作られたプログラムが多くの人にとって使い良い道具となる好例である．Moran のプログラムは英語用だったが，電子技術総合研究所の田中穂積，淵一博両氏によって日本語用に書き直され，東京大学の田中裕一氏によって推論機能が強化され，さらに電子技術総合研究所の佐藤泰介氏によって状景を作図する機能が組み込まれて現在に至っている．この作図には前述のグラフィック入出力機能が用いられている．

4. 将来の開発について

LISP 1.9 の開発は，従来もそうであったように二つの方向から進められるべきである．一つはシステム自体の機能強化であり，いま一つは応用プログラムの開発である*.

LISP 1.9 の現状について，ライブラリ中のグラフィック入出力を組み込んだり，引数の処理機能を CONNIVER レベルに引き上げたり，ブロックのコントロールを開放したりしてはどうかという意見もあるが，現在主たる応用分野として自然言語処理をかかえているために，当面の課題としては，

1) 入出力機能の拡充——ランダム・ファイルに対する入出力や，漢字，ひらがな等の日本語出力，またユーザがバッファを管理した一括入出力などを行う．
2) ストリング処理機能の拡充——ストリングのデータ表現そのものを改善し，部分ストリングを使い易くすると共に，アレイ—ストリングの変換機能や，ストリング入力機能を追加する．

が，最大の急務となっている．(システム自体を更新しなくてはどうにもならない．)

さらに，処理速度の向上（コンパイラ！）やトレース機能の向上が当面の課題となっている．

将来，上述の意見等をいれて，より使いやすいシステムにしたいと願っている．

5. あ と が き

LISP 1.9 という名称は，LISP 1.6 作成後，3年に渡る（光陰矢の如し！）改版作業を示している．(何故か LISP 4.6 にはならなかった．) LISP 1.9 は多くのユーザ，主として電子技術総合研究所のユーザの協力によって現状にまで到達できた．改めて感謝する次第である．特に推論機構研究室の淵一博室長，田中穂積，横井俊夫氏は，外部仕様作成から始まって製作中にも，様々の示唆，批判を与えて下さった．ここで紹介した応用プログラムのほとんどは推論機構研究室で開発されたものである．LISP 1.9 の入出力機能については東京大学の玉木久夫氏の助力を得た．最後に，拙文に紹介することを快諾して下さった LISP 1.9 の応用プログラムの作成者の方々に重ねて感謝の意を申し陳べる．

* 一般に LISP のような研究開発の最前線にある言語ほど，その言語を用いる応用プログラムとの関係が深く，言語自身の独り歩きは，時によって不幸な結果を招くことになりかねない．

参 考 文 献

1) 淵一博，田中穂積：言語理解システム開発のための道具立て——LISP を中心とするファシリティ群——，情報処理学会計算言語学研究会資料，CL 4-2, (Dec. 1975).
2) 後藤英一，金田康正：記号及び数式処理システム HLISP-REDUCE, ，情報処理学会記号処理研究委員会資料，(Nov. 1975).
3) 後藤典孝，田中穂積，淵一博：INFORMATION ABOUT LINGOL，電子技術総合研究所，(Jun. 1975).
4) 黒川利明，八木正博：TOSBAC-5600 LISP 1.6, 東芝レビュー，Vol. 29, No. 10, (Oct. 1974).
5) 黒川利明：TOSBAC-5600 LISP 1.6 システム，情報処理学会第 15 回大会，306, (Dec. 1974).
6) 黒川利明：LISP のデータ表現——TOSBAC-5600 LISP を中心にして——，情報処理，Vol. 17, No. 2 (Feb. 1976).
7) T. Kurokawa: A New Fast and Safe Marking Algorithm, (in preparation).
8) T. Kurokawa: New LISP Interpreter, (in preparation).
9) D. V. McDermott, G. J. Sussman: The Conniver Reference Manual, MIT AI Memo 259a, (Jan. 1974).
10) D. A. Moon: MACLISP Reference Manual, Project MAC-MIT, (Apr. 1974).
11) T. P. Moran: MILISY: the MIni-LInguistic System, CMU, (1974).
12) 長尾真，辻井潤一：自然言語処理のためのプログラミング言語 PLATON，情報処理，Vol. 15, No. 9, pp. 654～661, (Sep. 1975).
13) V. R. Pratt: A Linguistic Oriented Programming Language, Proc. 3rd IJCAI, pp. 372～381, (Aug. 1973).
14) V. R. Pratt: LINGOL——A Progress Report, Proc. 4 IJCAI, pp. 422～428, (Sep. 1975).
15) G. J. Sussman, T. Winograd, E. Charniak: MICRO-PLANNER REFERENCE MANUAL, MIT AI Memo 203A, (1971).
16) 田中穂積：統語情報を用いて文のわかち書きを行なうプログラム，情報処理学会第 16 回大会，(Nov. 1975).
17) 田中穂積，佐藤泰介，田中裕一：実験用日本語質問応答システム——MILISYJ——，情報処理学会第 16 回大会，(Nov. 1975).
18) 田中穂積，佐藤泰介，淵一博：日本語質問応答小規模実験システムについて，電子通信学会 AL 研究会資料，AL 75-48, (Nov. 1975).
19) W. Teitelman: INTERLISP Reference Manual, Xerox, (Feb. 1974).
20) T. Winograd: Procedures as a Represertation for Data in a Computer Program for Understanding Natural Language, Ph. D. Thesis MIT,

(1971).
21) LISP 1.6 User's Manual, EPICS-5-ON, 電子技術総合研究所, (Mar. 1974).
22) LISP 1.9 User's Manual, EPICS-5-ON-2, 電子技術総合研究所, (Mar. 1976).
23) micro-PLANNER User's Manual, EPICS-13-ON, 電子技術総合研究所, (Oct. 1974).
24) 小さな日本語質問応答システム MILISYJ プログラム説明書, 電子技術総合研究所・推論機構研究室内部資料, (Oct. 1975).
25) 記号処理シンポジウム報告集, (Aug. 1974).

(昭和51年2月17日受付)
(昭和51年4月1日再受付)