ACOS-LISPの機能拡張とMicro-ELIの移植

石　原　好　宏*

An Extension of ACOS-LISP and Transfer of Micro-ELI

Yoshihiro ISHIHARA

Abstract

One of the plausible methods to bridge the gap between the two LISP dialects is to define, as a part of the basic software, a group of functions in the dialect already implemented on the machine at hand. This technique is well-known and used among the skilled LISP users. But it is, in practice, a considerably troublesome job to manage, at least for the beginners, for the lack of their experience and appropriate materials.

This report describes the author's attempt of expanding the descriptive power of ACOS-LISP, and that of transferring the UCI-LISP version of MCELI to the extended ACOS-LISP system, which was done as a temporary goal of the extension. All of the LISP functions introduced in the course of the attempt are also presented. Many of them had been originally defined by Schank, R. C. et al., but had to be fixed in most cases in transferring them. And the rest had to be newly added by the author.

1. まえがき

20年余り前にMcCarthy, J. らが初めて開発したLISP[1]は，開発当初よりむしろ最近になって強い関心が寄せられるようになり，記号処理・数式計算・人工知能などの分野で今日広く利用されている．LISPの普及・発展には，FORTRAN, COBOL, PL/Iなどとは逆の要因が見られる[2]．後者の場合は国内的・国際的に言語仕様が標準化されたことが大きな要因となっているのに対して，前者の場合は言語仕様の標準化が行われず，多様な方言を個性的に発展させる土壌が作られたことが大きな要因と考えられている．その土壌とは，処理系の作成者とユーザとの間に存在した密接な情報交換の習慣である．もっとも，最近ではユーザ層の変化もあって，そのような状況が必ずしも十分には期待できない．そうなるとユーザは，当面のLISP方言の作成(implementation)に特有な情報を何らかの方法で入手することができ，他方LISP処理系の動作，組込関数ごとの振舞などに関する基礎知識を十分に持ち合せていない限り，新しい方言をすぐに使いこなすことは難しい．筆者は最近そのような経験をした．それは筆者が，オペレーティング・システムACOS-6の下で稼動するLISP処理系LISP 1.5/TSS-6 Release No. 3.0（これをACOS-LISPと呼ぶ）の処理能力を拡張して，UCI-LISPで書かれた実習用の英語文解釈システムMicro-ELI[3]（以後MCELIと呼ぶ）を移植しようとしたときの経験である．その際利用できたACOS-LISPの資料は，組込関数ごとに引数と関数値ないし副作用に関する簡単な説明を付けただけのものであった[4]．この外にACOS-LISPのRelease No. 2.0のプログラミング説明書[5]はあったものの，言語仕様の変更が大幅であったため，もはや信頼できる資料としては使えなかった．従ってACOS-LISP処理系の具体的振舞は，そのつどテスト・プログラムを作って動かしてみる外なかった．

ここで筆者が痛感したのは，一つのLISP処理系を作成した場合作成者は，プログラミングに関連のある作成データをユーザに伝える義務があるということであった．

本稿では，筆者が行ったACOS-LISPについての一つの試みを上述の観点から整理する．まず，LISPを使ったプログラムの作成と移植の手法を概観する．次にACOS-LISPの機能拡張の概要とその作業の過

* 工業短期大学部情報処理工学科

程で分かった ACOS-LISP の特徴をまとめる．次いで，機能拡張した ACOS-LISP 上に MCELI を移植する段階で新しく発見できた ACOS-LISP の特徴，移植の概要などをまとめる．そして移植した MCELI の動作実験についても述べる．なお，今回の機能拡張のために，そのままあるいは修正を加えて導入した既存の関数群[3),6),7)]，及び新たに定義した関数群は付録にまとめる．

2. LISP 環境下でのプログラム開発と移植

すでに何らかの LISP 処理系が手もとにある場合，プログラムを新しく開発したり，他の LISP 処理系から移植する方法として次の三つが考えられる．

方法1) 処理内容やユーザの好みに合う LISP 処理系を新たに作成し，その環境下で仕事をする：——その場合，プログラムを新しく開発することは容易となろう．また，移植すべきプログラムに対する修正も必要ないであろう．なぜならば，既存の処理系とは関係なく，新しい要求を満たす処理系が作られるからである．しかしそのような処理系を作成すること自体には相当の時間・労力・知識を必要とする．

方法2) 手近かにある計算機上で稼動中の LISP 処理系がある場合，その組込関数群の記述能力を変えずに仕事をする：——この場合，その LISP 方言の記述能力の違いにより，ユーザの負担はかなり違ってくる．ACOS-LISP の場合基礎が LISP 1.5 であるため，その記述能力を変えずに応用プログラムを書くのは不便かつ煩雑である．

方法3) 手もとの LISP 処理系に支援関数群を付加し，所望の記述能力を等価的に作り上げてから仕事をする：——この場合，既存の組込関数を再定義しない限り，プログラムの開発の際，従来の LISP 方言による言語仕様，及び支援関数群によって変化した言語仕様のどちらも自由に使える．これは LISP 本来の性質に基づく[6),7),9)]．従って，支援関数群の工夫次第では，比較的少ない時間と労力で，使い易い処理系を実現することができる．

今回は方法3を採用した．それは，既存の LISP 処理系ならば直ちに利用できること，そして比較的簡単にその機能が拡張でき，他の方言で書かれたプログラムの移植も比較的容易に行えること，などによる．

3. ACOS-LISP の機能拡張

3.1 作業の方針

方法3を採る場合でも，根本的に機能を拡張したければ，処理系の内部に直接手を加える必要がある．しかし，センター・システムのような場合，それを安易に行うわけにはいかない．従ってここでは LISP の関数定義機能を使った拡張を試みる．

ところで，LISP の使い易さと記述能力は，利用できる関数定義用及び制御用の関数に大きく依存する．そこで当面は，UCI-LISP で書かれた MCELI[3)] が最少限の修正で ACOS-LISP 上へ移植できることを一応の目安として関数の追加を試みる．そのため，Schank, R. C. らが示した54個の基本関数[10)] は極力そのままの仕様で使えるように努める．

以上の方針でこれまでに implement した関数群を，主な用途別に整理して次節に示す．各関数の定義式，その出所等の詳細は付録を参照願いたい．

3.2 基本関数の増強

(1) 関数定義用関数——EXPR 関数定義用に DE, FEXPR 関数定義用に DF, MACRO 関数定義用に DM を導入した．また，定義済みの EXPR 関数を処理系から除去する関数 UNDEFINE も導入した．但し，readmacro 定義用の DRM は定義していない．

(2) 制御用関数——まず，n 個の S 式を順次評価する関数として PROG1 と PROGN を定義した．ここで，PROG1 は最初の S 式の評価値を，また PROGN は最後のそれをそれぞれの関数値とする．さらに，COND 関数によく似た SELECTQ, MAP 関数の MAPC, MAPCAN, 繰返し制御用の LOOP, FOR, SOME も導入した．また，LAMBDA 変数に初期値を与えた後で PROGN の振舞をする LET も導入した．

(3) リスト処理用関数——対リストを探索する ASSOC, リストの長さを調べる LENGTH, リストの末尾に S 式を cons する CONS-END, 空リストにリストを cons する NCONS, 束縛変数*1) に set された pushdown stack を操作する PUSH と POP, APVAL 定数*2) に cset された pushdown stack を操作する PUSHC と POPC を定

*1) LAMBDA 変数または PROG 変数.
*2) 大域変数とも呼ぶ.

義した．この外，n 個のリストを append する MULTIAPPEND も加えた．

(4) 属性値処理用関数——アトムに属性と属性値を与える PUTPROP と DEFPROP を導入した．後者は，3つの引数がいずれも評価されない点で前者と異なる．

(5) 述語関数——S 式がアトムのとき NIL，さもなければ S 式そのものを返す CONSP と，指定した S 式がリストのトップレベルで見つかったときそれ以降の部分リストを返す IN-LIST，の2つを定義した．

以上の外に，54個の基本関数[10] の中には出力用関数として MSG と SPRINT があったが，今回は除外した．なお，各関数の定義に伴って定義した補助関数については言及しなかった．これについては付録を参照願いたい．

3.3 ACOS-LISP の特徴——UCI-LISP との相異点——（その1）

UCI-LISP の言語仕様と異なるために，3.2 の基本関数群を定義する際問題となった ACOS-LISP の特徴に次のようなものがある．

(1) FEXPR 関数の仮引数は2個である：

その第一引数には実引数全体を一つのリストとしたものが渡される．一方，第二引数には，この関数が評価されるときの結合リスト（＝そのときの環境）が渡される．ところが，このようなことは ACOS-LISP の添付資料等[4),5)] には明記されていない．しかも，UCI-LISP や LISP 1.9[7)] では第二引数そのものが不要である．従って，この事実に気づくのには時間がかかった．

(2) 評価関数 EVAL の第二引数として結合リストが必要である：

UCI-LISP で例えば，

```
(EVAL (EQUAL *WORD* 'TO))
```

と書くところを，ACOS-LISP では，

```
(EVAL (EQUAL *WORD* 'TO) A-LIST)
```

と書く必要がある．ここに，A-LIST は，*WORD* が束縛変数のときには，結合リストに束縛されていなければならない．ところが，*WORD* が APVAL 定数のときには，評価の際結合リストを見る必要がないので A-LIST のところを NIL としてよい．なお，今のところ筆者は，FEXPR 関数の中を除いて，結合リストを EVAL 関数の中から参照する方法を知らない．

(3) LAMBDA 式の実行部を構成する S 式は1個である：

UCI-LISP の場合，n 個の S 式を許す．従って，UCI-LISP で書かれたプログラムが LAMBDA 式を含み，かつその実行部が複数個の S 式から成るならば，その全体を PROGN でくくるように基本関数を修正する必要がある．

この外，FEXPR 型として定義した関数で変数の衝突による異常動作を経験した．これは LISP に共通の問題である[6)]．FEXPR 型関数を定義するときは，仮引数の命名に特別の注意が必要である．

4. MCELI の移植

3. に述べた基本関数群の追加によって，基本的にはほとんど無修正で MCELI を稼動させる準備が整った．しかし実際には，なお幾つかの組込関数及び処理系の振舞に微妙な差異があり，それを吸収するための MCELI 自体の修正も若干必要であった．

そのような微妙な差異を ACOS-LISP の側からながめ，MCELI の修正の概要を示す．

4.1 ACOS-LISP の特徴——UCI-LISP との相異点——（その2）

(1) 代入関数を2種類もっている：

〔実状〕 UCI-LISP では，束縛変数への代入も，大域変数への代入も SET（または SETQ）関数が使われる．これに対して ACOS-LISP では，SET（または SETQ）は束縛変数に対してだけ使われ，大域変数への代入は CSET（または CSETQ）関数が使われる．

〔対策〕 変数がソース・プログラムの中に陽に現れる場合，それをどちらとして取扱うべきかの判断は文脈から簡単に行える．しかし単語辞書の中に随時使われる変数の場合，今のところ EVAL 関数の中から結合リストが自由には参照できない*1) ため APVAL 定数として処理せざるを得ない．

(2) 空リストに対する CAR, CDR の振舞が違う：

〔現象〕 UCI-LISP の場合，どちらも NIL を返すようである*2)．ところが ACOS-LISP では，

*1) 3.3 の (2) を参照のこと．

*2) UCI-LISP の処理系もそれについての詳細な説明書も手もとにないため，MCELI のソース・プログラムの書き方から推定した．

CAR に対して ***** を，CDR に対して (APVAL (NIL) PNAME (*****)) を返す．ここで ***** は，ポインタが存在するものの，それがシステムの管理下に置かれておりユーザに見せたくないときに使われているようである．従ってこのようなデータを得た後そのまま処理を続けようとすると，遠からず MEMORY FAULT を引き起す．

〔対策〕 UCI-LISP 処理系が正しく処理する S 式例えば，

```
(SETQ *WORD* (POP *SENTENCE*))
```

を ACOS-LISP 処理系で処理すると，変数 *SENTENCE* が空リストになった時点でこの問題に直面する[*1]．これを避けるためには，上の S 式を，

```
(AND (CSETQ *WORD* *SENTENCE*)
     (CSETQ *WORD* (POPC *SENTENCE*)))
```

とすればよい．

(3) 実引数と仮引数の個数が一致しないときの処置が違う：

〔実状・対策〕 例えば制御用関数 LOOP の補助関数 LOOP1 の UCI-LISP による定義式[10] の中に，

```
(CONS 'RETURN R-BODY)
```

という部分がある．この S 式は R-BODY が空リストのとき (RETURN) となる．これが正しく処理される以上，UCI-LISP の処理系は対応する実引数の存在しない仮引数には NIL を既定値として与え，そのまま処理を続ける機能をもっていると考えられる．

ACOS-LISP 処理系では，実引数と仮引数が一致しない場合直ちにエラーメッセージを出し，処理を中止する．従って上例のような S 式が正常に処理されるためには，例えば，

```
(COND (R-BODY (CONS 'RETURN R-BODY))
      (T (LIST 'RETURN R-BODY)))
```

と書き換える必要がある．

4.2 MCELI の修正作業

基本関数を増強 (3.2 参照) してもなお幾つかの点で修正が必要であった．そのことについて簡単に述べる．なお，3.4 と 4.1 に述べたことから容易に推察できるものは一部省略する．

(1) UCI-LISP 版で PROG 変数として扱われている *WORD* を大域変数 (ここでは APVAL 定数) に変更した．これは，3.3 の(2)に述べたように，PROG 変数のままでは EVAL による評価ができないためである．

(2) UCI-LISP 版の関数 RUN-STACK の中に含まれている S 式，

```
(SETQ REQUEST (CHECK-TOP *STACK*))
```

を，

```
(AND *STACK*
     (CSETQ REQUEST (CHECK-TOP *STACK*)))
```

と書き換えた．これは，CHECK-TOP が *STACK* の car をとる関数であることから，*STACK* が空リストになっても異常動作を起さず正常処理を続けるための措置である．なお，ACOS-LISP Release No. 3.0 の AND は，UCI-LISP の場合と同じく実引数を左から順次調べていく．そして全引数が非 NIL の場合，最後の引数の評価値を返す．

(3) UCI-LISP の出力関数 MSG が ACOS-LISP では未定義であるため，当面は PRONG, TERPRI, PRINT の組合せで内容だけはともかく出力できるようにした．

なお，MCELI が使う単語辞書への単語データの登録は，単語ごとに関数 DEF-WORD を使って行う (付録参照)．単語データは Packet 形式[3] で記述するが，その構文は修正する必要がなかった．

5. MCELI の動作実験

3. と4. に述べた一連の作業で，MCELI の ACOS-LISP 上への移植はひとまず終わった．そこで，今回の試みで採用した MCELI の動作環境の構成例を示し，MCELI の動作実験の一部を紹介する．

5.1 MCELI の動作環境

今回の試みでは，MCELI を ACOS-LISP 上に移植することを一応の作業目標とした．しかしその意図するところは，ACOS-LISP の処理系に UCI-LISP の基本関数群[10] をできる限りそのままの言語仕様で取込み，ACOS-LISP の機能拡張を図ることであった．そこで，新しく導入する関数群は今後の使い勝手も考えて，次の 4 つのファイルに分類・格納した．

*1) POP, POPC が CAR を含んでいる (付録参照)．

(1) AUGLISP0

3.2に述べた基本関数群のうち，AUGLISP1 に登録した LOOP, FOR, LET を除く全部とそれらの補助関数群を登録したファイルである．

(2) AUGLISP1

このファイルには，制御用関数の LOOP, FOR, LET とそれらの補助関数群をまとめた．

(3) MCELIPRC

MCELI を構成する関数群のうち，処理手続きと単語データ登録のための関数群だけをまとめた．

(4) MCELIDAT

MCELI のうち，単語データを定義する部分だけをまとめた．

関数群をこのように分類・整理しておけば，それらを処理系へ組込む場合，増強・修正等を行う場合など，作業を能率よく進めることができる．

なお，MCELIDAT に納めた単語辞書は，当初 Schank, R. C. が与えた 9 単語 (LISP のソースで約60行)[3] だけであったが，その後27単語 (同じく約250行) に増強した．

```
>
>(PROCESS-TEXT '((BETTY WENT TO HIM)))

 INPUT IS --*-
(BETTY WENT TO HIM)

 PROCESSING WORD IS --*-
*START*

 PROCESSING WORD IS --*-
BETTY


**
*PART-OF-SPEECH* --*-
NOUN-PHRASE

*CD-FORM* --*-
(PERSON (NAME (BETTY)))

*SUBJECT* --*-
(PERSON (NAME (BETTY)))

 PROCESSING WORD IS --*-
WENT

*PART-OF-SPEECH* --*-
VERB

*CD-FORM* --*-
(PTRANS (ACTOR (*VAR* GO-VAR1)) (OBJECT (*VAR* GO-VAR1)) (TO (*VAR*
  GO-VAR2)) (FROM (*VAR* GO-VAR3)))

GO-VAR1 --*-
(PERSON (NAME (BETTY)))

GO-VAR3 --*-

**
(LOCATION (OF (PERSON (NAME (BETTY)))))

*CONCEPT* --*-
(PTRANS (ACTOR (*VAR* GO-VAR1)) (OBJECT (*VAR* GO-VAR1)) (TO (*VAR*
  GO-VAR2)) (FROM (*VAR* GO-VAR3)))

 PROCESSING WORD IS --*-
TO

 --*- NOT IN THE DICTIONARY

 PROCESSING WORD IS --*-
HIM

*PART-OF-SPEECH* --*-
NOUN-PHRASE

*CD-FORM* --*-
(PERSON (SEX (MALE)) (REF (DEF)))

GO-VAR2 --*-
(PERSON (SEX (MALE)) (REF (DEF)))


**
 CD FORM IS --*-
(PTRANS (ACTOR (PERSON (NAME (BETTY)))) (OBJECT (PERSON (NAME (BETTY))))
  (TO (PERSON (SEX (MALE)) (REF (DEF)))) (FROM (LOCATION (OF (PERSON (
  NAME (BETTY)))))))
VAL..
NIL
   348 MSEC.

>
```

Fig. 2 A log of the MCELI, applied to the sentence "*Betty went to him.*"

5.2 MCELI の動作実験と結果

MCELI を ACOS-LISP 上で動かすためには，5.1 に示した各ファイルの関数群を順次，処理系に組込む必要がある．そのための手順を ACOS-6 のコマンド列で Fig. 1 に示す．

```
SYSTEM ?LISP

LISP1.5/TSS-6 R3.0
COMPILE?NO



**

>/NARG
>/TYPE MIXED
>/NVAL
>/NTIM
>/RUN AUGLISP0
>/RUN AUGLISP1
>/RUN MCELIPRC
>/RUN MCELIDAT
>/PTIM
>/PVAL
>
```

Fig. 1 A command sequence for setting up the MCELI.

この操作を完了すると MCELI が起動可能な状態となる．そこで英語文 *Betty went to him.* を解釈したときの経過と結果を Fig. 2 に示す．

この例文以外にも，単語辞書に登録済みの単語を使って作れる多くの英語文の解釈を試み，いずれも正しく処理していることを確かめた．

6. むすび

手近かに使える LISP 処理系の関数定義用関数を使えば，自分が必要とする他の処理系の記述能力を比較的手軽に手もとの LISP 処理系に取込むことができることを，ACOS-LISP 処理系を使って確認した．なお，ファイル AUGLISP0 と AUGLISP1 に納めた基本関数群は，ACOS-LISP のような LISP1.5 レベルの方言を機能拡張したい場合，まず組込むべき関数群であると言える．

この試みには山口大学情報処理センターの ACOS 800 を利用した．その際すべての作業は，我々の研究室のマイクロコンピュータ IF-800 モデル20を知能端末[*1] として使い，構内公衆回線経由で行ったことを付記しておく．

参考文献

1) McCarthy, J. et al.: "LISP 1.5 programmar's manual", MIT press (1962).
2) 黒川利明："LISP", 情報処理, 22, 6, 469-472 (1981)
3) Schank, R. C. et al.: "Inside computer understanding: Five programs plus miniatures", Lawrence Erlbaum Associates, Hillsdale, New Jersey (1981).
4) 日本電気(株)："LISP 1.5/TSS-6 R3.0 添付資料" (1982)
5) 日本電気(株)："ACOS オペレーティングシステム ACOS プログラム管理 LISP 1.5 プログラミング説明書", FGM 01-2 (1981)
6) Chaniak, E. et al.: "Artificial intelligence programming", Lawrence Erlbaum Associates, Hillsdale, New Jersey (1980).
7) 黒川利明："LISP 入門", 培風館 (1982)
8) Laurent Siklóssy (後藤, 戸島訳)："LISP 入門", 日本コンピュータ協会 (1981)
9) Winston, P. H. et al.: "LISP", Addison-Wesley (1981)
10) 文献 3) の pp. 54-60 参照

付 録

ACOS-LISP の機能拡張，あるいは MCELI の移植のために今回導入した全関数の定義式をファイル別にまとめて示す．なお，各関数の出所と修正の有無を明らかにするために，それぞれの定義式の先頭行の左側に記号 x：y を付した．ここに x は出所を示す．すなわち，S のとき文献3)，C のとき文献6)，K のとき文献7)，L のとき文献8) からの引用であることを，また I のとき筆者が独自に定義したものであることを示す．一方，y は，各関数の引用に当って筆者が修正ないし変更を行っていれば I，原版のままを引用していれば O とした．

(昭和 58 年 4 月 14 日 受理)

[*1] そのための知能端末プログラムは，当時本学科勤務であった渡部哲夫氏（現，付属図書館医学部分館整理係）が BASIC で作成したものである．

```
      >/LISTH AUGLISP0

         FILE NAME =  auglisp0     04/08/83     17:39:42

      0010DEFINE ((
 I:0  0020  (DE (LAMBDA (FNAME ARGS BODY)
      0030    (DEFLIST (LIST (LIST FNAME (LIST 'LAMBDA ARGS BODY))) 'EXPR) ))
 I:0  0040  (DF (LAMBDA (FNAME LARG BODY)
      0050    (DEFLIST (LIST (LIST FNAME (LIST 'LAMBDA LARG BODY))) 'FEXPR) ))
      0060  ))
 I:0  0070DE (PUTPROP (ATM EXP IND)
      0080     (PROG2 (DEFLIST (LIST (LIST ATM EXP)) IND) (LIST ATM IND)) )
 0:I  0090DF (DEFPROP (&&L& *A&) (PUTPROP (CAR &&L&)(CADR &&L&)(CADDR &&L&)))
 I:0  0100DF (PROGN (%%%L%%% A%) (PROG (%%X%% &LL%)
      0110         (SETQ &LL% %%%L%%%)
      0120      L1 (SETQ %%X%% (CAR &LL%)) (SETQ &LL% (CDR &LL%))
      0130         (COND ((NULL &LL%)(RETURN (EVAL %%X%% A%))))
      0140         (EVAL %%X%% A%)
      0150         (GO L1) ))
 I:0  0160DF (PROG1 (%&L&% A*) (PROG ($X1& #LL&)
      0170         (SETQ $X1& (EVAL (CAR %&L&%) A*))
      0180         (SETQ #LL& %&L&%)
      0190      L1 (SETQ #LL& (CDR #LL&))
      0200         (COND ((NULL #LL&)(RETURN $X1&)))
      0210         (EVAL (CAR #LL&) A*)
      0220         (GO L1) ))
 S:I  0230DF (DM (##L# A**) (PROGN
      0240      (PUTPROP (CAR ##L#) (CONS 'LAMBDA (CDR ##L#)) 'MACRO)
      0250      (PUTPROP (CAR ##L#)
      0260               (LIST 'LAMBDA
      0270                     '(%%%L A-LIST)
      0280                     (LIST 'MACHAC (LIST 'QUOTE (CAR ##L#)) '%%%L) )
      0290               'FEXPR )
      0300      (LIST (CAR ##L#) 'MACRO) ))
 S:I  0310DE (MACHAC (%%%FN %%%L)
      0320      (EVAL ((ISMACRO %%%FN)(CONS %%%FN %%%L)) A-LIST) )
 0:0  0330DE (ISMACRO (FN) (AND (ATOM FN)(NOT (NUMBERP FN))(GET FN 'MACRO)))
 S:I  0340DE (DEFFN (L TYPE)
      0350      (LIST 'DEFPROP
      0360            (CADR L)
      0370            (EXPANDMACROS
      0380              (CONS 'LAMBDA
      0390                    (COND ((NULL (CDR (CDDDR L)))
      0400                           (CONS (CADDR L) (CDDDR L)) )
      0410                          (T (LIST (CADDR L)
      0420                                   (CONS 'PROGN (CDDDR L)) )) ) ) )
      0430            TYPE ) )
 S:0  0440DE (EXPANDMACROS (X)
      0450      (COND ((OR (ATOM X) (EQUAL (CAR X) 'QUOTE)) X)
      0460            ((GET (CAR X) 'MACRO)
      0470               (EXPANDMACROS ((GET (CAR X) 'MACRO) X)) )
      0480            (T (CONS (EXPANDMACROS (CAR X))
      0490                     (EXPANDREST (CDR X)) )) ) )
 S:0  0500DE (EXPANDREST (L)
      0510      (COND ((ATOM L) L)
      0520            (T (CONS (EXPANDMACROS (CAR L))
      0530                     (EXPANDREST (CDR L)) )) ) )
      0540REMPROP (DE EXPR)
      0550REMPROP (DF EXPR)
 S:0  0560DM (DE (L) (DEFFN L 'EXPR))
 S:0  0570DM (DF (L) (DEFFN L 'FEXPR))
 I:0  0580DM (CONSP (EXP*)
      0590      (LIST 'COND (LIST (LIST 'ATOM (CADR EXP*)) 'NIL)
      0600                  (LIST 'T (CADR EXP*)) ) )
 I:0  0610DE (ASSOC (IDENT L) (PROG (LT LR)
      0620         (SETQ LR L)
      0630      L1 (SETQ LT (CAR LR))
      0640         (SETQ LR (CDR LR))
      0650         (COND ((EQUAL (CAR LT) IDENT)(RETURN LT))
      0660               ((NULL LR) (RETURN NIL)) )
      0670         (GO L1) ))
 S:0  0680DE (CONS-END (L X) (APPEND L (LIST X)))
 I:0  0690DF (SELECTQ (%L% **A**) (PROG (#X1#* &LL&&)
      0700         (SETQ #X1#* (EVAL (CAR %L%) **A**))
      0710         (SETQ &LL&& (CDR %L%))
      0720      L1 (COND ((NULL (CDR &LL&&)) (RETURN (EVAL (CAR &LL&&) **A**)))
      0730               ((OR (AND (ATOM (CAAR &LL&&))
      0740                         (EQUAL #X1#* (CAAR &LL&&)) )
      0750                    (AND (NOT (ATOM (CAAR &LL&&)))
      0760                         (MEMBER #X1#* (CAAR &LL&&)) ) )
      0770                (RETURN (SELECTQ-AUX (CDAR &LL&&))) ) )
      0780         (SETQ &LL&& (CDR &LL&&))
      0790         (GO L1) ))
 I:0  0800DE (SELECTQ-AUX (L%) (PROG (LL)
      0810         (SETQ LL L%)
      0820      L1 (COND ((NULL (CDR LL))(RETURN (EVAL (CAR LL) **A**))))
      0830         (EVAL (CAR LL) **A**)
      0840         (SETQ LL (CDR LL))
      0850         (GO L1) ))
```

Fig. 3 The content of AUGLISP0.

```
S:0  0860DM (PUSH (L) (LIST 'CAR
     0870                    (LIST 'SETQ
     0880                          (CADDR L)
     0890                          (LIST 'CONS (CADR L) (CADDR L)) ) ))
S:I  0900DM (PUSHC (L) (LIST 'CAR
     0910                     (LIST 'CSETQ
     0920                           (CADDR L)
     0930                           (LIST 'CONS (CADR L) (CADDR L)) ) ))
S:0  0940DM (POP (L) (LIST 'PROG1
     0950                  (LIST 'CAR (CADR L))
     0960                  (LIST 'SETQ (CADR L)
     0970                        (LIST 'CDR (CADR L)) ) ))
S:I  0980DM (POPC (L) (LIST 'PROG1
     0990                   (LIST 'CAR (CADR L))
     1000                   (LIST 'CSETQ (CADR L)
     1010                         (LIST 'CDR (CADR L)) ) ))
I:0  1020DE (MAPCAN (LIS FUN)
     1030     (COND ((NULL LIS) ())
     1040           (T (APPEND (FUN (CAR LIS)) (MAPCAN (CDR LIS) FUN))) ) )
L:0  1050DE (MAPC (LIS FUN) (PROG (LL)
     1060         (SETQ LL LIS)
     1070     L1 (COND ((NULL LL) (RETURN NIL)))
     1080        (FUN (CAR LL))
     1090        (SETQ LL (CDR LL))
     1100        (GO L1) ))
I:0  1110DE (SOME (LIS FUN) (PROG (LL)
     1120         (SETQ LL LIS)
     1130     L1 (COND ((NULL LL) (RETURN NIL))
     1140              ((FUN (CAR LL)) (RETURN LL)) )
     1150        (SETQ LL (CDR LL))
     1160        (GO L1) ))
I:0  1170DE (IN-LIST (IND LST) (PROG (R)
     1180         (SETQ R LST)
     1190     L1 (COND ((NULL R) (RETURN NIL))
     1200              ((EQUAL IND (CAR R)) (RETURN R)) )
     1210        (SETQ R (CDR R))
     1220        (GO L1) ))
K:0  1230DM (MULTIAPPEND (L)
     1240     (COND ((NULL (CDR L)) NIL)
     1250           ((NULL (CDDR L)) (CADR L))
     1260           (T (LIST 'APPEND (CADR L)
     1270                    (CONS (CAR L) (CDDR L)) )) ) )
 .
 0:I 1280DF (UNDEFINE (**#L*# *#A) (UNDEFINE-LIST **#L*#))
 0:I 1290DE (UNDEFINE-LIST (L)
     1300     (MAPC L (FUNCTION (LAMBDA (X) (REMPROP X 'EXPR)))) )
S:0  1310DE (LENGTH (L) (COND ((NULL L) 0)
     1320                     (T (PLUS 1 (LENGTH (CDR L)))) ))
I:0  1330DE (NCONS (L) (CONS L ()))
```

Fig. 3 Continued.

```
>/LISTH AUGLISP1

   FILE NAME =  auglisp1    04/08/83    17:49:44

S:0 0010DM (LOOP (L) (LOOP1 (CDR L)
    0020                    (GET-KEYWORD 'INITIAL L)
    0030                    (GET-KEYWORD 'RESULT L) ))
S:I 0040DE (LOOP1 (CLAUSES I-BODY R-BODY)
    0050     (MULTIAPPEND
    0060       (LIST 'PROG (VAR-LIST I-BODY))
    0070       (SETQ-STEPS I-BODY)
    0080       (CONS 'LOOP (MAPCAN CLAUSES 'DO-CLAUSE))
    0090       (LIST '(GO LOOP)
    0100             'EXIT
    0110             (COND (R-BODY (CONS 'RETURN R-BODY))
    0120                   (T      (LIST 'RETURN R-BODY)) ) ) ) )
S:I 0130DE (GET-KEYWORD (KEY L) (PROG (X)
    0140     (SETQ X (ASSOC KEY (CDR L)))
    0150     (COND ((NULL X) (RETURN NIL))
    0160           (T (RETURN (CDR X))) ) )))
S:I 0170DE (DO-CLAUSE (CLAUSE)
    0180     (SELECTQ (CAR CLAUSE)
    0190       ((INITIAL RESULT) NIL)
    0200       (WHILE (LIST (LIST 'OR (CADR CLAUSE) '(GO EXIT))))
    0210       (DO    (CDR CLAUSE))
    0220       (UNTIL (LIST (LIST 'AND (CADR CLAUSE) '(GO EXIT))))
    0230       (PROGN (TERPRI)
    0240              (PRIN1 '$$$ UNKNOWN KEYWORD --*- $)
    0250              (PRINT CLAUSE)
    0260              (TERPRI) ) ) )
```

Fig. 4 The content of AUGLISP1.

```
S:0  0270DE (VAR-LIST (L)
     0280      (COND ((NULL L) NIL)
     0290            (T (CONS (CAR L) (VAR-LIST (CDDR L)))) ) )
S:0  0300DE (SETQ-STEPS (L)
     0310      (COND ((NULL L) NIL)
     0320            ((NULL (CADR L)) (SETQ-STEPS (CDDR L)))
     0330            (T (CONS (LIST 'SETQ (CAR L) (CADR L))
     0340                     (SETQ-STEPS (CDDR L)) )) ) )
S:0  0350DM (FOR (L)
     0360      (FOR1 (CADR L)
     0370        (GET-KEYWORD 'WHEN   L)
     0380        (GET-KEYWORD 'DO     L)
     0390        (GET-KEYWORD 'SAVE   L)
     0400        (GET-KEYWORD 'EXISTS L) ) )
S:I  0410DE (FOR1 (IN WHEN DO SAVE EXISTS)
     0420      (CONS (FOR-MAPFN WHEN DO SAVE EXISTS)
     0430            (APPEND (CDDR IN)
     0440                    (LIST (FOR-LAMBDA (CAR IN)
     0450                                      WHEN DO SAVE EXISTS )) ) ) )
S:0  0460DE (FOR-MAPFN (WHEN DO SAVE EXISTS)
     0470      (COND (DO     'MAPC)
     0480            (EXISTS 'SOME)
     0490            (WHEN   'MAPCAN)
     0500            (T      'MAPCAR) ) )
S:I  0510DE (FOR-LAMBDA (VAR WHEN DO SAVE EXISTS)
     0520      (LIST 'FUNCTION
     0530            (CONS 'LAMBDA
     0540                  (CONS (LIST VAR)
     0550                        (FOR-LAMBDA-AUX WHEN DO SAVE EXISTS) ) ) ) )
S:I  0560DE (FOR-LAMBDA-AUX (WHEN DO SAVE EXISTS)
     0570      (PROG (BODY)
     0580          (SETQ BODY (COND (WHEN (FOR-WHEN WHEN DO SAVE))
     0590                           (T    (OR DO SAVE EXISTS)) ))
     0600          (RETURN  (COND ((CDR BODY) (LIST (CONS 'PROGN BODY)))
     0610                         (T          BODY) )) ) )
S:0  0620DE (FOR-WHEN (WHEN DO SAVE)
     0630      (LIST
     0640        (LIST 'COND
     0650              (APPEND (ADD-PROGN WHEN)
     0660                      (OR DO
     0670                          (LIST (CONS 'NCONS (ADD-PROGN SAVE))) ) ) )) )
S:0  0680DE (ADD-PROGN (L)
     0690      (COND ((CDR L) (LIST (CONS 'PROGN L)))
     0700            (T       L) ) )
S:0  0710DM (LET (L) (LET1 (REVERSE (CADR L)) NIL NIL (CDDR L)))
S:I  0720DE (LET1 (L VARS VALS BODY)
     0730      (COND ((NULL L)
     0740             (CONS (CONS 'LAMBDA
     0750                         (COND ((CDR BODY)
     0760                                (LIST VARS (CONS 'PROGN BODY)) )
     0770                               (T (CONS VARS BODY)) ) ) VALS) )
     0780            (T (LET1 (CDDR L)
     0790                     (CONS (CADR L) VARS)
     0800                     (CONS (CAR L)  VALS)
     0810                     BODY )) ) )
```

Fig. 4 Continued.

```
    )/LISTH MCELIPRC

       FILE NAME =  mceliprc    04/08/83    18:14:53

S:I  0010 DE (PROCESS-TEXT (TEXT)
     0020      (FOR (SENTENCE IN TEXT)
     0030        (DO (PROGN (TERPRI)
     0040                   (PRINT '$$$ INPUT IS --*- $)
     0050                   (PRINT SENTENCE) )
     0060            (LET (CD (PARSE SENTENCE))
     0070                 (PROGN (TERPRI)
     0080                        (PRINT '$$$ CD FORM IS --*- $)
     0090                        (PRINT CD) ) ) ) ) )
S:I  0100 DE (PARSE (SENTENCE)
     0110      (CSETQ *CONCEPT* NIL)(CSETQ *WORD* NIL)
     0120      (CSETQ *STACK* NIL)  (CSETQ *SENTENCE* (CONS '*START* SENTENCE))
     0130      (LOOP
     0140        (WHILE  (AND   (CSETQ *WORD* *SENTENCE*)
     0150                       (CSETQ *WORD* (POPC *SENTENCE*)) ))
     0160        (DO     (PROGN (TERPRI)
     0170                       (PRINT '$$$ PROCESSING WORD IS --*- $)
     0180                       (PRINT *WORD*) )
     0190                (LOAD-DEF)
     0200                (RUN-STACK) )
     0210        (RESULT (REMOVE-VARIABLES *CONCEPT*)) ) )
```

Fig. 5 The content of MCELIPRC.

```
S:I 0220 DE (RUN-STACK ()
     0230      (LOOP (INITIAL REQUEST NIL TRIGGERED NIL)
     0240        (WHILE  (AND *STACK* (SETQ REQUEST (CHECK-TOP *STACK*))))
     0250        (DO     (POPC *STACK*)
     0260                (DO-ASSIGNS REQUEST)
     0270                (PUSH REQUEST TRIGGERED) )
     0280        (RESULT (ADD-PACKETS TRIGGERED)) ) )
S:O 0290 DE (IS-TRIGGERED (REQUEST)
     0300      (LET (TEST (REQ-CLAUSE 'TEST REQUEST))
     0310        (OR (NULL TEST) (EVAL (CAR TEST) ())) ) )
S:O 0320 DE (DO-ASSIGNS (REQUEST)
     0330      (LOOP (INITIAL ASSIGNMENTS (REQ-CLAUSE 'ASSIGN REQUEST))
     0340        (WHILE  ASSIGNMENTS)
     0350        (DO     (REASSIGN (POP ASSIGNMENTS) (POP ASSIGNMENTS))) ) )
S:I 0360 DE (REASSIGN (VAR VAL)
     0370      (COND ((CSET VAR (EVAL VAL ()))
     0380             (PROGN (TERPRI)
     0390                    (PRIN1 VAR)(PRIN1 '" --*- ")(TERPRI)
     0400                    (PRINT (EVAL VAR ()))  ) )) )
S:O 0410 DE (ADD-PACKETS (REQUESTS)
     0420      (FOR (REQUEST IN REQUESTS)
     0430        (DO   (ADD-STACK (REQ-CLAUSE 'NEXT-PACKET REQUEST))) ) )
S:I 0440 DE (REMOVE-VARIABLES (CD-FORM)
     0450      (COND ((ATOM CD-FORM) CD-FORM)
     0460            ((IS-VAR CD-FORM)
     0470               (REMOVE-VARIABLES (EVAL (NAME:VAR CD-FORM) ())) )
     0480            (T (LET (VAL NIL)
     0490                 (CONS (HEADER:CD CD-FORM)
     0500                       (FOR (PAIR IN (ROLES:CD CD-FORM))
     0510                         (WHEN (SETQ VAL (REMOVE-VARIABLES
     0520                                          (FILLER:PAIR PAIR) )))
     0530                         (SAVE (LIST (ROLE:PAIR PAIR) VAL)) ) ) )) ) )
S:O 0540 DE (IS-VAR       (X) (AND (CONSP X) (EQUAL (CAR X) '*VAR*)))
S:O 0550 DE (NAME:VAR     (X) (AND (CONSP X) (CONSP (CDR X)) (CADR X)))
S:O 0560 DE (HEADER:CD    (X) (CAR X))
S:O 0570 DE (ROLES:CD     (X) (CDR X))
S:O 0580 DE (ROLE:PAIR    (X) (CAR X))
S:O 0590 DE (FILLER:PAIR (X) (CADR X))
S:O 0600 DE (TOP-OF       (STACK)  (CAR STACK))
S:I 0610 DE (ADD-STACK    (PACKET) (AND  PACKET (PUSHC PACKET *STACK*))  PACKET)
S:I 0620 DE (LOAD-DEF ()
     0630      (LET (PACKET (GET *WORD* 'DEFINITION))
     0640        (COND (PACKET (ADD-STACK PACKET))
     0650              (T (PROGN
     0660                   (TERPRI)
     0670                   (PRINT '$$$ --*- NOT IN THE DICTIONARY $)) ) ) ) )
S:O 0680 DE (REQ-CLAUSE (KEY L)
     0690      (LET (X (ASSOC KEY L))  (AND X (CDR X))) )
S:I 0700 DF (DEF-WORD (&&&&L*  A-LIST)
     0710      (PUTPROP (CAR &&&&L*)(CDR &&&&L*) 'DEFINITION) (CAR &&&&L*) )
I:O 0720 DE (FEATURE (CD-FORM PRED)
     0730      (EQUAL (CAR CD-FORM) PRED) )
I:O 0740 DE (GET-NP-PREDICTION (STACK)
     0750      (LET (PACKET (TOP-OF STACK))
     0760        (LOOP (INITIAL FEATURE-ARGS NIL  EQUAL-ARGS NIL
     0770                       FEATURE-ARG2 NIL  REQUEST    NIL
     0780                       TEST-CLAUSE  NIL  TEST       NIL )
     0790          (WHILE  (AND PACKET  (SETQ REQUEST (POP PACKET))))
     0800          (UNTIL  (NOT (AND (SETQ TEST (REQ-CLAUSE 'TEST REQUEST))
     0810                            (EQUAL 'AND (CAAR TEST))
     0820                            (SETQ TEST-CLAUSE  (CDAR TEST))
     0830                            (SETQ EQUAL-ARGS
     0840                                  (REQ-CLAUSE 'EQUAL TEST-CLAUSE) )
     0850                            (EQUAL ''NOUN-PHRASE (CADR EQUAL-ARGS))
     0860                            (SETQ FEATURE-ARGS
     0870                                  (REQ-CLAUSE 'FEATURE TEST-CLAUSE) )
     0880                            (SETQ FEATURE-ARG2 (CADR FEATURE-ARGS)) )))
     0890          (RESULT FEATURE-ARG2) ) ) )
I:O 0900 DE (GET-CD-FORM (REQUEST)
     0910      (LOOP (INITIAL ASSIGN (REQ-CLAUSE 'ASSIGN REQUEST)
     0920                     VAR NIL  VAL NIL )
     0930        (WHILE  (AND ASSIGN
     0940                     (SETQ VAR (POP ASSIGN))
     0950                     (SETQ VAL (POP ASSIGN)) ))
     0960        (UNTIL  (EQUAL VAR '*CD-FORM*))
     0970        (RESULT (EVAL VAL ())) ) )
I:O 0980 DE (RESOLVE-CONFLICT (REQUESTS)
     0990      (LOOP (INITIAL REQUEST NIL  CD-FORM NIL)
     1000        (WHILE  (AND REQUESTS
     1010                     (SETQ REQUEST (POP REQUESTS)) ))
     1020        (UNTIL  (FEATURE (GET-CD-FORM REQUEST)
     1030                         (EVAL *PREDICTED* NIL) ))
     1040        (RESULT REQUEST) ) )
I:O 1050 DE (CHECK-TOP (STACK)
     1060      (COND
     1070        (STACK
     1080          (LET (PACKET (TOP-OF STACK)  TRIG-REQS NIL)
     1090            (SETQ TRIG-REQS
     1100                  (FOR (REQUEST IN PACKET)
     1110                    (WHEN (IS-TRIGGERED REQUEST))
     1120                    (SAVE REQUEST) ) )
     1130            (COND ((NULL TRIG-REQS) NIL)
     1140                  ((CDR TRIG-REQS) (RESOLVE-CONFLICT TRIG-REQS))
     1150                  (T (CAR TRIG-REQS)) ) ) ) ) )
```

Fig. 5 Continued.

```
)/LISTH MCELIDAT

   FILE NAME =  mcelidat    04/08/83    18:25:26



S:I  0250 DEF-WORD (WENT
     0260   ((ASSIGN *PART-OF-SPEECH* 'VERB
     0270            *CD-FORM*          '(PTRANS (ACTOR  (*VAR* GO-VAR1))
     0280                                        (OBJECT (*VAR* GO-VAR1))
     0290                                        (TO     (*VAR* GO-VAR2))
     0300                                        (FROM   (*VAR* GO-VAR3)) )
     0310            GO-VAR1    *SUBJECT*
     0320            GO-VAR2    NIL
     0330            GO-VAR3    (LIST 'LOCATION (CONS 'OF (NCONS *SUBJECT*))) )

     **
     0340    (NEXT-PACKET
     0350       ((TEST (EQUAL *WORD* 'TO))
     0360        (NEXT-PACKET
     0370          ((TEST   (EQUAL *PART-OF-SPEECH* 'NOUN-PHRASE))
     0380           (ASSIGN GO-VAR2 *CD-FORM*) ) ) )
     0390       ((TEST   (EQUAL *WORD* 'HOME))
     0400        (ASSIGN GO-VAR2 '(HOUSE)) ) ) ) )


S:I  0690 DEF-WORD (*START*
     0700   ((ASSIGN *PART-OF-SPEECH* NIL
     0710            *PREDICTED*       NIL
     0720            *CD-FORM*         NIL
     0730            *SUBJECT*         NIL
     0740            *PREDICATES*      NIL )
     0750    (NEXT-PACKET
     0760      ((TEST   (EQUAL *PART-OF-SPEECH* 'NOUN-PHRASE))
     0770       (ASSIGN *SUBJECT* *CD-FORM*)
     0780       (NEXT-PACKET
     0790         ((TEST   (EQUAL *PART-OF-SPEECH* 'VERB))
     0800          (ASSIGN *CONCEPT* *CD-FORM*) ) ) ) ) ) )


I:O  1640 DEF-WORD (BETTY
     1650   ((ASSIGN *PART-OF-SPEECH* 'NOUN-PHRASE
     1660            *CD-FORM*         '(PERSON (NAME (BETTY))) )) )


I:O  1730 DEF-WORD (HIM
     1740   ((ASSIGN *PART-OF-SPEECH* 'NOUN-PHRASE
     1750            *CD-FORM*         '(PERSON (SEX (MALE)) (REF (DEF))) )) )
```

Fig. 6 The content of MCELIDAT (a portion, necessary to perform the interpretation shown in Fig. 2).