ACOS-LISPの機能拡張（その2）

石 原 好 宏*

An Extension of ACOS-LISP (Part II)

Yoshihiro ISHIHARA*

Abstract

Functions for doing input and output tend to be given short shrift in LISP, and this is the case with ACOS-LISP as well. But those functions are important especially when one intends to build a system in which user interaction is needed and information needs to be displayed legibly. As for output functions used for editing, the original version of ACOS-LISP has essentially only one built-in function, PRIN1, that moves information to the output buffer. But PRIN1 deals only with atoms. So I intend to augment ACOS-LISP with output functions to edit any type of S-expressions by making use of the output buffer.

Prior to it, however, I had to define several basic stringmanipulation functions to treat the S-expressions, because they are also lacking.

Thus, this report gives first a brief analysis of bad circumstances of original ACOS-LISP with respect to string manipulation and then gives the basic functions to cope with it. Introduction of several output functions to ACOS-LISP follow together with several other useful functions, such as for data-type definition, data-driven programming, list processing, and controlling. Definitions of the newly introduced functions are also given, though not in full.

1. まえがき

LISP は“豊かな拡張可能性をもち，極端に簡素化された基盤言語である”と言う意味で最初のエレガントな言語である[1]．しかしこのことは，使用目的に合わせた適切な拡張をしない限り，LISP が決して使い勝手のよい言語ではありえないことを示唆している．しかるに LISP には標準の言語仕様がないので，それを拡張するときに遭遇する問題は，個々の LISP システム，それを走らせるオペレーティング・システム，拡張の内容などによって異なる場合が少なくない．従って個々の拡張作業の内容を整理しておくことは，同じ努力を繰返す無駄を省くためにも必要と考える．

筆者は最近オペレーティング・システム ACOS-6 の下で稼動する LISP1.5/TSS-6 Release No. 3.0（以下 ACOS-LISP）の機能拡張を行い，その概要の一部を前稿[2)*1)]にまとめた．本稿では，その後に整備した文字列処理と出力処理の機能についてまずまとめる．さらに，知的な情報処理システムの記述に有用であると考えて新たに定義あるいは移植した関数群についても簡単に整理する．最後に，こうして収集した関数群を ACOS-LISP に組込んでで き上った一つの新しい LISP システムの使用方法についても簡単に言及する．なお，今回の機能拡張のために新たに定義あるいは移植した関数のうち主なものの定義式は付録にまとめる．

2. 機能拡張の背景と方針

前稿では機能拡張の当面の目標を，UCI-LISP で書かれた Micro-ELI[3] ができる限り少ない修正で ACOS-LISP の上に移植できることに置いて，基本的な関数群の定義と移植を試みた[2]．しかし前稿の段階で出力処理の機能を増強することまではできなかった．その理由は，これらの関数を定義するときに必要な，データとしての S 式（＝文字の連鎖）を処理する機能（これを本稿では文字列処理機能*1) と呼ぶ）をもつ基本関数が不備であり，それを直ちに定義すること

* 工業短期大学部情報処理工学科

*1) 以下の本文で“前稿”は文献 2) を指す．

*1) 本稿で文字列処理という場合，一表現形式としての文字列 (3.1 参照) の処理と，それ以外の単純な文字連鎖（リスト・データもその一つ）と考えられデータの処理の両方を含意する．

も筆者の手に余ったからである.

しかし，筆者のこのたびの機能拡張の一つの目標は，ACOS-LISP の上に一つの知的な自然語処理システムを構築することである．この立場からも，データとしての S 式を処理する基本関数を増強する必要がある．

そこで本稿では，まず ACOS-LISP の文字列処理機能の現状を分析し，追加すべき文字列処理機能を検討した後，実際に定義または移植した文字列処理用，及び出力処理用の関数を整理する．また，知的情報処理システムを構築するための道具の一つとして導入したデータ型定義用，データ駆動型プログラミング用などの関数，並びに前稿の範囲内の関数であるがその後に追加したものなどについても簡単に整理しておく．なお，本稿で試みた機能拡張も，前稿と同じく，LISP の関数定義機能を利用した[2]．

3. 文字列処理機能の拡張

データとしての S 式に対する文字列処理機能としてここでは，S 式を文字の連鎖と見たときの長さ（＝構成文字数）の測定，S 式の文字オブジェクトへの分解，その逆処理などを考える．

3.1 ACOS-LISP の文字列処理機能の現状

まず指摘すべきことは，文字列 (string) とその処理の概念が，ACOS-LISP では必ずしも明確でないことである．確かに，文字列の表現形式を "$l_1 \ldots l_n$"，すなわち両端を二重引用符で囲んだ30文字以内の任意のつづりとすることが明記され，ATOMSLENGTH, PRINTS など幾つかの文字列処理関数を備えている[4]．しかし，文字列は文字アトムの一表現形式としてしか扱われておらず，識別子 (identifier) としての文字アトムとの違いが判然としない．文字列を識別する関数 STRINGP などはもちろん組込まれていない．STRINGP を定義しようとすれば文字列の特徴である両端の二重引用符の存在を調べる必要があるが，その手立ては今のところない．筆者は，組込み済みの関数の振舞から文字列処理の底流にあるはずの論理構造を探り出そうとした．しかし，いまのところ有意なものは見出せていない．

このような状況下でとりあえず可能な改良は，アトムに対してしか許していない文字列処理を，リストに対しても許すことである．この立場から既存の基本関数の機能を調べていくと，次のような問題点が指摘できる．

(1) アトムを分解してその構成文字のリストを作る関数はある*1) が，任意の S 式について同じ処理をする関数がない．

(2) n 個の文字オブジェクトを n 個の引数として受取って一つの S 式を作る関数はある*2) が，n 個の文字オブジェクトを一つのリストとして受取って S 式を作る関数がない．

(3) アトムの長さを調べる関数はある*3) が，任意の S 式の長さを調べる関数がない．

以上のことから我々は三つの基本関数を定義した．その概要を次節にまとめる．

3.2 基本関数の増強

任意の S 式に対する文字列処理は次の方針で実現できる．S 式がアトムのときは既存の基本関数を使う．S 式がリストのときは，それを左右一対のカッコとその中味とに分けて，中味（＝リストの要素）を順次処理する．そのとき要素がアトムであれば，S 式がアトムの問題に帰着できる．要素が再びリストであれば再帰的に S 式がリストのときの処理をすればよい．なお，この処理で，リストの中の要素と要素の間隔は，原データの実情に関係なく，すべて区切り記号（ここでは空白）一個分に正規化して取扱う．

この方針に基づいて定義した各関数のタイプ，機能，処理上の特徴などを次にまとめる．

（1） EXPLODE (EXPR 関数)

〔定形〕 (EXPLODE S-exp) ここに S-exp は任意の S 式．

〔機能〕 任意の S 式を分解し，その構成文字のリストを返す．

〔処理〕 EXPLODE が引数として受取る S 式がアトムならば，直ちに基本関数 UNPACK で処理する．一方，S 式がリストのときは，その両端のカッコとリストの中味とに分離し，後者の処理を補助関数 EXPLODE-AUX (EXPR 関数) に任せる．補助関数では，リストの要素を一つずつ取出し，それを再帰的に EXPLODE で順次処理し，その結果を append していく．こうして得られる最終結果を最終の関数値として戻す．

（2） READLIST (EXPR 関数)

---

*1) UNPACK

*2) PACK（これの各引数は文字オブジェクト．任意個の引数をとれる．）

*3) ATOMSLENGTH と ATOMLENGTH（前者は文字列の中にアトムの区切り記号を含むとき文字列の両端の二重引用符を含めた長さ，さもなければ二重引用符の有無に関係なくそれを除いた長さを返す．後者は，二重引用符の有無に関係なくそれを除いた長さを返す．）

〔定形〕 (READLIST C-list) ここに C-list は文字オブジェクトのリスト.

〔機能〕 n 個の文字オブジェクトを一つのリストで受取り, 一つの S 式を作って返す.

〔処理〕 この関数も最後の処理は基本関数 PACK に任せる. ただし, リスト形式で受取ったデータを関数 PACK の引数とするために, 前処理として, 補助関数 READLIST-AUX (EXPR 関数) でリストの各文字オブジェクトに QUOTE をかける. 得られたリストを主関数の中でアトム PACK と cons し, さらにその結果を評価して最終結果を得る.

(3) SEXPLENGTH (EXPR 関数)

〔定形〕 (SEXPLENGTH S-exp) ここに S-exp は任意の S 式.

〔機能〕 任意の S 式の長さを調べる. ただし, リスト内の要素間の間隔はすべて長さ 1 とみなす.

〔処理〕 S式がアトムならば直ちに ATOMLENGTH で処理する. リストの場合は EXPLODE で分解した後 LENGTH で調べる.

以上の基本関数を使えば幾つかの便利な出力処理用の関数を定義できる. 新たに定義または移植した出力処理用の関数については次章で整理する.

4. 出力処理機能の拡張

4.1 ACOS-LISP の出力処理機能の現状

ACOS-LISP に初めから組込まれた出力処理機能で当面の機能拡張に必要なものは概略次のとおりである. ただし, ここではデータの出力ないし編集に直接関連する機能だけに注目する. また, ここでは TSS 端末から LISP システムを操作する場合だけを念頭に置く.

(1) 任意のS式が, 関数 PRINT あるいは PRINTS を使って出力できる. この場合, データは直ちに指定した出力ファイルへ送出され, 改行コード EOR も挿入される. 両関数の唯一の違いは, C クラスの文字*1) を含む文字列形式のデータのとき, PRINTS だけが文字列の両端を " で囲んで出力することである. その外の場合, 筆者の試みた限りでは, 両者とも文字列は " で囲まれることもなく, 違いが認められない*2).

(2) 出力データを, 直ちに出力ファイルに送出するのでなく, 出力バッファ*1) まで転送したいときは関数 PRIN1 または PRIN1S を使う. ただし, これらの関数では出力データをアトムに限る. 二つの関数の違いは(1)の場合と同じである.

(3) 出力ファイルとして, TSS 端末とパーマネント・ファイルのどちらでも任意に選ぶことができる.

(4) 現在出力中の行について, 出力バッファが保持している文字数を関数 PRINTCOLUMN-COUNT で調べることができる.

(5) 出力バッファが保持する内容を, 指定の出力ファイルへ送出するためには関数 TERPRI を使う. TERPRI が評価されると出力バッファの中は空白で初期化され, 次に出力される先頭文字位置は 1 となる.

上記の機能をそのままにして, 読み易い出力画面を作るプログラムを直ちに書くのは難しい. データを, 出力ファイルに送出する前に, 出力バッファ上で編集できる機能が必要である. そのために改良すべき点は次の二つに要約できる.

(1) 出力画面上の出力文字位置, 出力文字数などが自由に管理できること.

(2) 出力バッファへ転送可能なデータのタイプを, アトムに限らず, リストを含めた任意の S 式に拡張すること.

以上の検討に基づいてこのたび六つの基本関数と二つの実用関数を定義した. その概要は次節にまとめる.

4.2 基本関数と実用関数の増強

初めに六つの基本関数のそれぞれについて, 使い方, 機能, 処理の特徴などを整理する. 六つの関数はいずれも FEXPR 型である. これは引数の省略を許すためである. 引数の省略は, 各定形の〔 〕の内側全部を同時に省略するときに限って許す. また, 省略した場合の既定値は, 出力ファイル f_n が TSS 端末, その外については個別に示す. なお, 各関数の引数は評価される.

(1) CUR-COL

〔定形〕 (CUR-COL [f_n])

〔機能〕 次に転送する文字列の先頭文字の, 現在出力中のファイル f_n の出力バッファ上の文字位置を返す.

---

*1) ACOS-LISP では使用可能な文字種を6つのクラスに分けている[4]. Cクラスには, (,), [,], , , , , 空白, ', EOR, EOF が属する.

*2) ACOS-LISP では文字列の概念が明確でない と 3.1 で述べた根拠の一つがこれである.

*1) 添付資料[4]に明記されてはいないが, 出力ファイルの 1 レコード分の長さを持つと考えらる.

〔処理〕 基本関数 PRINTCOLUMNCOUNT を使って容易に定義できる.

(2) CHARCT

〔定形〕 (CHARCT [f_n])

〔機能〕 ファイル f_n の出力バッファに出力可能な文字数を返す.

〔処理〕 CUR-COL を使って定義できる. なお, ここで1レコード長は72文字 (=TSS 端末への標準最大出力文字数[4]) とする.

(3) SPACES

〔定形〕 (SPACES [sp[f_n]]), ここに sp は挿入したい空白の文字数. 既定値は 1.

〔機能〕 sp で指定した文字数の空白をファイル f_n の出力バッファに転送する. ただし, 転送の途中で出力バッファが72文字を越えたら, その時点で転送は打切る.

〔処理〕 CUR-COL と PRIN1 を使って定義できる.

(4) LINES

〔定形〕 (LINES [l_n[f_n]]), ここに l_n は挿入したい空白行の行数. 既定値は 0.

〔機能〕 ファイル f_n 上に現在出力中の行と, 次に文字列を出力する予定の行との間に l_n 行の空白行を挿入する. ただし, 未送出の文字列が出力バッファ上にあるときは, それを出力ファイルへ送出した後で l_n 行の空白行を挿入する.

〔処理〕 CUR-COL と TERPRI を使って定義する.

(5) TAB

〔定形〕 (TAB [tb[f_n]]), ここに tb はタブセットする文字位置. 既定値は 1.

〔機能〕 ファイル f_n の出力バッファ上に次に転送する文字列の先頭文字位置 tb となるように設定する.

〔処理〕 出力バッファの現在の文字位置 (CUR-COL f_n) が tb に等しいときは何もしない. tb より小さければ, それが tb となるまで空白文字を埋める. tb より大きければ, 現在の出力バッファの内容をそのまま直ちに f_n へ送出し, その後改めて出力バッファの文字位置が tb となるように空白文字を埋める.

(6) PRIN1E

〔定形〕 (PRIN1E S-exp [tb[l_n[f_n]]]), ここに S-exp はファイル f_n の出力バッファに転送する任意の S 式. tb は, S-exp の出力中に改行するときの各行の先頭文字位置で, 既定値が 1. l_n は, S-exp を出力中の改行のとき, 間に挿入する空白行の行数で, 既定値が 0.

〔機能〕 任意の S 式をファイル f_n の出力バッファへ転送する. その意味で, PRIN1 の拡張版である. 従って, S-exp を出力バッファへ転送するときの最初の先頭文字位置は (CUR-COL f_n) とする. ただし, S-exp が長ければ, その出力中に改行が必要なことがある. そのような改行の行数と先頭文字位置はユーザが管理できるように配慮した.

〔処理〕 任意の S 式のうち, アトムは基本関数 PRIN1 で処理する. リストは, EXPLODE の場合と同様, 両端のカッコとその中味とに分けて処理する. 両端のカッコは PRIN1 で処理する. 中味は要素ごとに調べて, アトムならば PRIN1 で処理し, リストならば再帰的に上記の処理を繰返せばよい. なお, PRIN1S による処理は考えなかった.

以上六つの関数はいずれも筆者が独自に定義した. 続いて二つの実用関数について述べる. PRIN-LS は前稿の MCELI の出力形式を改良する過程で必要となって定義した. 一方, WRITE は文献 1) の定義式を一部修正して移植した.

(7) PRIN-LS (FEXPR 関数)

〔定形〕 (PRIN-LS L-sent[tb[l_n[f_n]]]), ここに L-sent はリストのリスト形式の文章データ. 例えば, ((I HAVE A BOOK ".") (I READ IT YESTERDAY ".")) である. tb は2行目以降のタブセット位置, l_n は次の出力行までの間に挿入する空白行の数. f_n は出力ファイル名.

〔機能〕 例えば上例の文章データが与えられると, この関数は **I HAVE A BOOK. I READ IT YESTERDAY.** と印刷する. ただし, 先頭行の先頭文字位置はユーザの責任で別に設定する必要がある. また, 途中で出力バッファが一杯になったときは自動的に出力ファイルにデータを送出する. ただし, 最終行では出力ファイルにデータが転送されるだけである. その後の処理は再びユーザの責任である.

〔特徴〕 LISP で書いた自然語処理システムでは, 文章を上記のようなリストのリスト形式で扱うことが多いと考える. 一方, 外部への出力の際はカッコを除去した形式の方がはるかに読み易い. この関数はその間げきを埋める有用性がある.

(8) WRITE[1] (MACRO 関数)

〔機能〕 基本的に文献 1) の WRITE と同じ仕様とした. ただし ACOS-LISP に文字列タイプがなく STRINGP が使えないため, アトムが単独に

現れたら，それを単なる文字列と見なして印刷するように変更した．同時に，タブ用の文字 T が文字列としてのアトムより前に検出できるように修正した．

以上により，出力処理に当面不便を来たさない程度の基本的機能は一応すべて準備できたことになる．

5. その他の処理機能の増強

ここでは3.と4.で扱わなかった関数の増強の概要を機能別に整理する．

5.1 高度の処理機能をもつ基本関数

ここでは二つのマクロ関数だけに注目する．

(1) データ型の定義，それによって定義されるデータ構造の各項目を参照する関数群の自動定義などを一挙に行う関数として RECORD-TYPE を文献 1) から移値した．ただし，定義式は Charniak, E. らが与えたものそのまま[1] ではない．データ・レコードにフラグを任意につけられる，そのフラグを手がかりとしたデータ識別関数が同時に定義できるなど，文献 1) に後出の各例題の実行に支障が出ない程度の改良を行っている．

(2) データ駆動型プログラミング用の関数として：＝を文献 1) から移植した．この関数も，原形のままではない．第二引数の中で特殊記号 *—* が使える，データ型定義関数で自動的に定義された関数も使える，などの改良を行っている．

5.2 その他の基本関数

(1) 制御用——FOR の定義式として前稿で Schank 版[3] を移植したが，その後 Charniak 版[1] に改めた．これは FOR macro の中で SPLICE, FILTER などの機能を使うためである．

(2) リスト処理と制御用——前稿で定義した MAPCAN の中の，処理結果を順次つなぐための関数 APPEND を NCONC に改めた．これは文献 1) の仕様に合わせるためである．この外，文献 1) の仕様に従って SUBSET, EVERY を新たに定義した．

(3) リスト処理用——EQ の意味で同じ要素がリスト内に含まれるか否かを調べる MEMQ，指定した要素と EQ の意味で同じ要素をリストから取除く DREMOVE，EQUAL の意味で同じ要素を取除く REMOVE の三つの関数を文献 5) から移植した．また，前稿で導入した ASSOC は，定義式はそのまま(連想リストの見出しを EQUAL で調べる）とし，関数名だけを ASSOC# に変えた．そして新たに，連想リストの見出しを EQ で調べるために ASSOC を定義した．これも UCI-LISP の仕様と一致させるためである．さらに，二つのリストの，EQUAL の意味での共通集合を求める INTERSECTION，任意個のリストについて同じ処理をする MULTI-INTERSECTION, MULTI-INTERSECTION-Q*[1] を定義した．

(4) 文字列処理用——二つの文字列をつないで一つの文字列（ただし30文字以下）にする CONCATENATE を新しく定義し，新しい文字アトムを次々に創り出す NEWSYM を文献 1) から移植した．

6. ACOS-LISP 拡張版の使い方及び検討

6.1 ACOS-LISP 拡張版の使い方

前稿と本稿にまとめた内容で ACOS-LISP の基本的拡張の目標点には到達したと考える．この先は，現在の状態を基盤にして，さらに問題の焦点を絞り込んで，より上位の関数群を目的別に作る必要がある．そこで，現時点を一つの区切りとして，これまで収集した関数群を組込んだ LISP システムを ACOS-LISP 拡張版と呼ぶことにする．

今回の一連の作業で収集した関数群を我々は次の三つのパーマネント・ファイルに分散格納して利用している．

（1） /LIB-LISP/AUGLISPA*[2]

この内容は，一部の修正や増補を除けば，前稿の AUGLISP0 と同じものである．

（2） /LIB-LISP/AUGLISPB

この内容は，全面的に移植し直した FOR マクロ関連の関数群を除いて，前稿の AUGLISP1 と同じものである．

（3） /LIB-LISP/AUGLISPC

このファイルには本稿で導入した文字列処理用，出力処理用の関数群全部の外，新たに定義ないし移植したその他の処理用の関数のほとんどを収めている．

このように分散格納するのは，もっぱらソフトウェアとしての保守管理の利便のためである．しかし，この場合システムの立上げには注意を要する．関数定義用の関数などでブートストラップを行う部分があるので，ACOS-LISP への関数組込みの順序をむやみに変

---

*1) マクロ関数の中で使うために，各リストに QUOTE をかける．

*2) LIB-LISP は ACOS-6 ファイル・システム[6] のサブカタログ名，AUGLISPA はファイル名である．

```
0010/NARG
0020/NTIM
0030/NVAL
0040/TYPE MIXED
0050/RUN /LIB-LISP/AUGLISPA
0060/RUN /LIB-LISP/AUGLISPB
0070/RUN /LIB-LISP/AUGLISPC
0080/PTIM
0090/PVAL
```

Fig. 1(a) The content of the permanent file, /SETUPCOM/LISPSET.

```
SYSTEM ?LISP

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


>/RUN /SETUPCOM/LISPSET

>
```

Fig. 1(b) Initial operation for setting up the extended version of ACOS-LISP.

えることは許さない．ここでは，(1)，(2)，(3)の順序でファイル内の関数が組込まれることを想定している．その場合 ACOS-6 のファイル・システムの下では次の方法をとればよい．あらかじめパーマネント・ファイルの中に Fig. 1(a) に示すコマンド列を収めておく．ACOS-LISP 拡張版の機能を使いたいときは，まず ACOS-LISP の原版を立上げ，その後で Fig. 1(b) に示す一つのコマンドを投入する．そうすれば，Fig. 1(a) に示すコマンド列が順次実行されて，その終了時には ACOS-LISP 拡張版が立上っている．

6.2 検　　討

ACOS-LISP では文字列の取扱いが必ずしも明確でないため，新しく導入した関数の定義式の中に筆者の誤解に起因する誤りがないとは言えない．しかし，これまで使用した限りでは特に不都合は生じていない．また，本稿で扱った範囲では，前稿の3.3と4.1で指摘した以外の問題点に出会うことはなかった．

このたびの機能拡張では LISP の関数定義機能という限られた手段だけを使った．それにもかかわらず ACOS-LISP 拡張版は，原版に比べて，多くの拡張機能を備え，相当に使い易くなっている．例えば，文献 1) の前半に紹介されているプログラム例などは，一部を除き，ほとんどそのままで ACOS-LISP 拡張版の上で正しく動くことを確認している．

7. むすび

今回の機能拡張では入力処理に全く手をつけなかった．それは，ACOS-LISP 原版がすでに文字，アトム，リストのデータを入力する関数をもっており，特に不便を感じなかったからである．しかし，内部コードそのままでの入力，READ マクロの定義などが可能になればさらに使い易い LISP システムになると思われる．しかし，このような機能は LISP システムの implementation に依存する部分が多く，関数定義機能だけで対処することはできない．

なお，以上の試みには山口大学情報処理センターの ACOS-800 を利用したことを付記する．

参 考 文 献

1) Charniak, E. et al.: "Artificial intelligence programming", Lawrence Erlbaum Associates, Hillsdale, New Jersey (1980)
2) 石原好宏："ACOS-LISP の機能拡張と Micro-ELI の移植"，山口大学工学部研究報告，**34**, 1, 113–123 (1983)
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 R 3.0 添付資料"(1982)
5) Meehan, J. R.: "The new UCI LISP manual", Lawrence Erlbaum Associates, Hillsdale, New Jersey (1979)
6) 日本電気(株)："ACOS オペレーティングシステム，ACOS-6 リモート処理タイムシェアリングシステム説明書", FEF 02-4 (1980)

付　　録

今回の機能拡張のために導入した関数のうち主なものの定義式を機能別にまとめて示す．なお，各定義式の出所と修正の有無を明示するために，各定義式の先頭行のコメント (/* の後の部分) に x:y の記号を付けた．x は出所を示し，x が I のとき筆者自身による定義，C のとき文献 1) からの引用であることを表す．一方，y は筆者による修正の有無を示し，それが I のとき拡張または修正あり，O のとき原版のままで修正がなかったことを表す．

(昭和 58 年 10 月 15 日　受理)

```
0110(DE EXPLODE (SEX)                                          /*  I:0  -58/06/25-
0120  (COND ((ATOM SEX) (UNPACK SEX))
0130        (T (MULTIAPPEND (LIST '"(")
0140                        (EXPLODE-AUX SEX)
0150                        (LIST '")") )) ) )
0160(DE EXPLODE-AUX (SEX)                                      /*  I:0  -58/06/25-
0170  (PROG (L R RSLT)
0180       (SETQ R SEX)
0190    L1 (SETQ L (CAR R))
0200       (SETQ R (CDR R))
0210       (SETQ RSLT (APPEND RSLT (EXPLODE L)))
0220       (COND ((NULL R) (RETURN RSLT)))
0230       (SETQ RSLT (APPEND RSLT (LIST '" ")))
0240       (GO L1) ) )
0290(DE SEXPLENGTH (SEX)                                       /*  I:0  -58/06/25-
0300  (COND ((ATOM SEX) (ATOMLENGTH SEX))
0310        (T (LENGTH (EXPLODE SEX))) ) )
0380(DE READLIST (CLST)                                        /*  I:0  -58/06/06-
0390  (EVAL (CONS 'PACK (READLIST-AUX CLST)) NIL) )
0400(DE READLIST-AUX (CLST)                                    /*  I:0  -58/06/06-
0410  (PROG (ANS)
0420    L1  (COND ((NULL CLST) (RETURN ANS))
0430              (T (SETQ ANS (APPEND ANS
0440                                   (LIST (LIST 'QUOTE (CAR CLST))) ))
0450                 (SETQ CLST (CDR CLST)) ) )
0460        (GO L1) ) )
0790(DE CONCATENATE (ATM1 ATM2)                                /*  I:0  -58/06/09-
0800  (READLIST (APPEND (EXPLODE ATM1) (EXPLODE ATM2))) )
```

Fig. 2 Functions for string manipulation.

```
0510(DF CUR-COL (#&$AG&& ALIS)                                 /*  I:0  -58/06/09-
0520  (ADD1
0530    (EVAL (CONS 'PRINTCOLUMNCOUNT
0540                (COND ((NULL #&$AG&&) (LIST ''#))
0550                      (T (LIST
0560                           (LIST 'QUOTE (EVAL (CAR #&$AG&&) ALIS)) )) ) )
0570          ALIS ) ) )
0850(DF SPACES (&%%EXP$ ALIS)                                  /*  I:0  -58/06/23-
0860  (PROG (N FL)
0870       (SETQ N   1)
0880       (SETQ FL '#)
0890       (COND ((NULL &%%EXP$) NIL)
0900             (T (SETQ N  (EVAL (CAR  &%%EXP$) ALIS))
0910                (COND ((NULL (CDR &%%EXP$)) NIL)
0920                      (T (SETQ FL (EVAL (CADR &%%EXP$) ALIS))) ) ) )
0930    L1 (COND ((OR (ZEROP N) (EQUAL (CUR-COL FL) 73)) (RETURN NIL))
0940             (T (PRIN1 '" " FL)
0950                (SETQ N (SUB1 N))
0960                (GO L1) ) ) ) )
1000(DF LINES (%%LN-FL& ALIS)                                  /*  I:0  -58/06/23-
1010  (PROG (LN FL)
1020       (SETQ LN  0)
1030       (SETQ FL '#)
1040       (COND ((NULL %%LN-FL&) NIL)
1050             (T (SETQ LN (EVAL (CAR %%LN-FL&) ALIS))
1060                (COND ((NULL (CDR %%LN-FL&)) NIL)
1070                      (T (SETQ FL (EVAL (CADR %%LN-FL&) ALIS))) ) ) )
1080       (COND ((NEQUAL (CUR-COL FL) 1) (TERPRI FL)))
1090    L1 (COND ((ZEROP LN) (RETURN NIL))
1100             (T (TERPRI FL)
1110                (SETQ LN (SUB1 LN)) ) )
1120       (GO L1) ) )
1170(DF TAB (%%%NS-FL& ALIS)                                   /*  I:0  -58/06/23-
1180  (PROG (NS FL N1 N2)
1190       (SETQ FL '#)
1200       (SETQ N1  1)
1210       (COND ((NULL %%%NS-FL&) NIL)
1220             (T (SETQ N1 (EVAL (CAR %%%NS-FL&) ALIS))
1230                (COND ((NULL (CDR %%%NS-FL&)) NIL)
1240                      (T (SETQ FL (EVAL (CADR %%%NS-FL&) ALIS))) ) ) )
1250       (SETQ N2 (CUR-COL FL))
1260       (SETQ NS (DIFFERENCE N1 N2))
1270       (COND ((MINUSP NS)
1280              (TERPRI FL)(SPACES (SUB1 N1) FL) )
1290             ((GREATERP NS 0) (SPACES NS FL))
1300             (T (RETURN NIL)) ) ) )
1350(DF CHARCT (%&FN&% AL)                                      /*  I:0  -58/06/13-
1360  (DIFFERENCE 73 (CUR-COL (COND ((NULL %&FN&%) '#)
1370                                (T (EVAL (CAR %&FN&%) AL)) ))) )
1440(DF PRIN-LS (&#S-C-L-F# AL)                                 /*  I:0  -58/09/19-
1450  (PROG (LS CL LF FN SEN)
1460       (SETQ CL 1) (SETQ LF 0) (SETQ FN '#)
1470       (SETQ LS (EVAL (CAR &#S-C-L-F#) AL))
1480       (COND ((NULL (CDR   &#S-C-L-F#)) NIL)
1490             (T (SETQ CL (EVAL (CADR  &#S-C-L-F#) AL))
```

Fig. 3 Functions for output processing

```
1500                (COND ((NULL (CDDR  &#S-C-L-F#)) NIL)
1510                      (T (SETQ LF (EVAL (CADDR &#S-C-L-F#) AL))
1520                         (COND ((NULL (CDDDR &#S-C-L-F#)) NIL)
1530                               (T (SETQ FN (EVAL (CAR (CDDDR &#S-C-L-F#))
1540                                                 AL ))) ) ) ) ) )
1550    L1 (COND ((NULL LS) (RETURN NIL))
1560             (T (SETQ SEN (CAR LS))
1570                (SETQ LS  (CDR LS)) ) )
1580    L2 (COND ((NULL SEN) (GO L1)))
1590       (COND ((OR (GREATERP (CHARCT FN) (SEXPLENGTH (CAR SEN)))
1600                  (EQUAL    (CHARCT FN) (SEXPLENGTH (CAR SEN))) ) NIL )
1610             (T (LINES LF FN)(TAB CL FN)) )
1620    L3 (PRIN1E (CAR SEN) CL LF FN)
1630       (COND ((AND (CDR SEN)(MEMQ (CADR SEN) '("," "." "?" "!"))) NIL)
1640             ((EQ (CAR SEN) '",")
1650              (COND ((EQUAL (CHARCT FN) 0) NIL)
1660                    (T (PRIN1 '" " FN)) ) )
1670             ((MEMQ (CAR SEN) '("." "?" "!"))
1680              (COND ((EQUAL (CHARCT FN) 0) NIL)
1690                    ((EQUAL (CHARCT FN) 1) (PRIN1 '" " FN))
1700                    (T (PRIN1 '"  " FN)) ) )
1710             (T (COND ((GREATERP (CHARCT FN) 0) (PRIN1 '" " FN)))) )
1720       (SETQ SEN (CDR SEN))
1730       (GO L2) ) )
1810(DF PRIN1E (#$S&C&L&F AL)                                  /*  I:0  -58/10/11-
1820  (PROG (SEX CL LF FN)
1830       (SETQ CL 1) (SETQ LF 0) (SETQ FN '#)
1840       (SETQ SEX (EVAL (CAR #$S&C&L&F) AL))
1850       (COND ((NULL (CDR #$S&C&L&F)) NIL)
1860             (T (COND ((NULL (CDDR #$S&C&L&F)) NIL)
1870                      (T (SETQ LF (EVAL (CADDR #$S&C&L&F) AL))
1880                         (COND ((NULL (CDDDR #$S&C&L&F)) NIL)
1890                               (T (SETQ FN (EVAL (CAR (CDDDR #$S&C&L&F))
1900                                                 AL ))) ) ) )
1910                (SETQ CL (EVAL (CADR #$S&C&L&F) AL)) ) )
1920       (COND ((NULL SEX) NIL)
1930             ((ATOM SEX) (COND ((OR (GREATERP (CHARCT FN)
1940                                              (ATOMLENGTH SEX) )
1950                                    (EQUAL    (CHARCT FN)
1960                                              (ATOMLENGTH SEX) ) )
1970                               (PRIN1 SEX FN) )
1980                              (T (LINES LF  FN)
1990                                 (TAB   CL  FN)
2000                                 (PRIN1 SEX FN) ) ))
2010             (T (PRIN1E-AUX SEX CL LF FN)) )
2020       (RETURN NIL) ) )
2030(DE PRIN1E-AUX (SEX CL LF FN)                              /*  I:0  -58/09/19-
2040  (PROG ()
2050       (PRIN1 '"(" FN)
2060   L1 (COND ((NULL SEX) (PRIN1 '")" FN)
2070                        (RETURN NIL) )
2080            ((ATOM (CAR SEX))
2090               (COND ((GREATERP (CHARCT FN) (SEXPLENGTH (CAR SEX)))
2100                      (PRIN1E (CAR SEX) CL LF FN)
2110                      (COND ((NULL (CDR SEX)) NIL)
2120                            (T (PRIN1 '" " FN)) ) )
2130                     ((EQUAL (CHARCT FN) (SEXPLENGTH (CAR SEX)))
2140                      (PRIN1E (CAR SEX) CL LF FN) )
2150                     (T (LINES LF FN)
2160                        (TAB CL FN)
2170                        (PRIN1 (CAR SEX) FN)
2180                        (COND ((NULL (CDR SEX)) NIL)
2190                              (T (PRIN1 '" " FN)) ) ) ) )
2200            (T (PRIN1E-AUX (CAR SEX) CL LF FN)
2210               (COND ((NULL (CDR SEX)) NIL)
2220                     (T (PRIN1 '" " FN)) ) ) )
2230      (SETQ SEX (CDR SEX))
2240      (GO L1) ) )
2260(DM WRITE (L)                                              /*  C:I  -58/06/23-
2270  (CONS 'PROGN
2280        (FOR (X IN (CDR L))
2290             (SAVE (COND ((EQ X 'T) '(TERPRI))
2300                         ((ATOM X)
2310                          (LIST 'PROGN (LIST 'PRIN1E (LIST 'QUOTE X))
2320                                       (LIST 'SPACES) ) )
2330                         ((AND (CONSP X)
2340                               (MEMQ (CAR X) '(LINES SPACES TAB)) )
2350                          X )
2360                         (T (LIST 'PROGN (LIST 'PRIN1E X)
2370                                         (LIST 'SPACES) )) )) ) ) )
```

Fig. 3 Continued

```
1370(DE SUBSET (LIS FUN)                                        /*  I:0  -58/07/15-
1380  (PROG (L)
1390    L1 (COND ((NULL LIS) (RETURN (REVERSE L)))
1400             ((FUN (CAR LIS)) (SETQ L (CONS (CAR LIS) L))) )
1410       (SETQ LIS (CDR LIS))
1420       (GO L1) ) )
1430(DE EVERY (LIS FUN)                                          /*  I:0  -58/09/12-
1440  (PROG ()
1450    L1 (COND ((NULL LIS) (RETURN T))
1460             ((FUN (CAR LIS)) (SETQ LIS (CDR LIS))
1470                              (GO L1) )
1480             (T (RETURN NIL)) ) ) )
1620(DE INTERSECTION (LST1 LST2)                                 /*  I:0  -58/08/01-
1630  (COND ((NULL LST1) NIL)
1640        ((MEMBER (CAR LST1) LST2)
1650         (CONS (CAR LST1) (INTERSECTION (CDR LST1) LST2)) )
1660        (T (INTERSECTION (CDR LST1) LST2)) ) )
1670(DM MULTI-INTERSECTION (L)                                   /*  I:0  -58/08/03-
1680  (COND ((NULL (CDR L)) NIL)
1690        ((NULL (CDDR L)) (CADR L))
1700        (T (LIST 'INTERSECTION
1710                 (CADR L)
1720                 (CONS (CAR L) (CDDR L)) )) ) )
1730(DM MULTI-INTERSECTION-Q (L)                                  /*  I:0  -58/08/03-
1740  (COND ((NULL (CDR L)) NIL)
1750        ((NULL (CDDR L)) (LIST 'QUOTE (CADR L)))
1760        (T (LIST 'INTERSECTION
1770                 (LIST 'QUOTE (CADR L))
1780                 (CONS (CAR L) (CDDR L)) )) ) )
```

Fig. 4 Functions for list processing and/or controlling.

```
2510(DM := (EXP)                                                /*  C:I  -58/06/28-
2520  (LET (LFT (CADR EXP) RGT (CADDR EXP) FORM NIL)
2530       (COND ((ATOM LFT) (LIST 'CSETQ LFT RGT))
2540             ((GET (CAR LFT) 'SET-PROGRAM)
2550              (CONS (GET (CAR LFT) 'SET-PROGRAM)
2560                    (APPEND (CDR LFT)
2570                            (LIST (SUBST LFT '*-* RGT)) ) ) )
2580             ((GET (CAR LFT) 'MACRO)
2590              (PROGN (SETQ FORM
2600                           (EXPANDMACROS ((GET (CAR LFT) 'MACRO) LFT)) )
2610                     (LIST ':= FORM RGT) ) ) ) ) )
2620(DEFPROP CAR RPLACA SET-PROGRAM)                             /*  C:O  -58/06/27-
2630(DEFPROP CDR RPLACD SET-PROGRAM)                            /*  C:O  -58/06/27-
2640(DEFPROP GET GET-SET-PROGRAM SET-PROGRAM)                   /*  C:O  -58/06/27-
2650(DE GET-SET-PROGRAM (ATM PROP VAL) (PUTPROP ATM VAL PROP)) /*C:O-58/06/27-
2660(DEFPROP CADR RPLACAD SET-PROGRAM)                          /*  I:O  -58/06/27-
2670(DE RPLACAD (EXP1 EXP2) (RPLACA (CDR EXP1) EXP2))           /*  C:O  -58/06/27-
2740(SPECIAL '(*TYPE* *FLGN*))                                  /*  C:I  -58/06/29-
2750(DM RECORD-TYPE (L)                                 /*  C:I  -58/06/29-58/07/07-
2760  (LET (*TYPE* (CADR L)
2770        *FLGN* (COND ((NULL (CDDDR L)) (CADR L))
2780                     ((EQ NIL (CADDR L)) NIL)
2790                     (T (CADDR L)) )
2800        SLOTS  (COND ((NULL (CDDDR L)) (CADDR L))
2810                     (T (CAR (CDDDR L))) ) )
2820       (EVAL (LIST 'DE
2830                   *TYPE*
2840                   (SLOT-FUNS-EXTRACT SLOTS NIL)
2850                   (COND (*FLGN* (LIST 'CONS
2860                                       (LIST 'QUOTE *FLGN*)
2870                                       (STRUC-CONS-FORM SLOTS) ))
2880                         (T (STRUC-CONS-FORM SLOTS)) ) ) NIL )
2890       (COND (*FLGN* (DEF-IS-*FLGN* *FLGN*))) ) )
2900(DE DEF-IS-*FLGN* (FLGN)                                     /*  I:O  -58/07/07-
2910  (EVAL (CONS 'DM
2920              (CONS (READLIST (APPEND (EXPLODE 'IS-)
2930                                      (EXPLODE *FLGN*) ))
2940                    (CONS '(L)
2950                          (LIST
2960                            (LIST
2970                              'LIST
2980                              ''AND
2990                              (LIST 'LIST ''CONSP '(CADR L))
3000                              (LIST 'LIST
3010                                    ''EQ
3020                                    (LIST 'QUOTE (LIST 'QUOTE FLGN))
3030                                    (LIST 'LIST
3040                                          ''CAR
3050                                          '(CADR L) ) ) ) ) ) ) ) ) NIL )
3060  NIL )
3070(DE SLOT-FUNS-EXTRACT (SLOTS PATH)                          /*  C:I  -58/06/29-58/07/05-
3080  (COND ((NULL SLOTS) NIL)
3090        ((ATOM SLOTS)
3100         (EVAL  (CONS 'DM
```

Fig. 5 Miscellaneous functions.

```
3110                        (CONS
3120                          (READLIST
3130                            (MULTIAPPEND
3140                              (EXPLODE SLOTS)
3150                              (LIST ':)
3160                              (EXPLODE *TYPE*) ) )
3170                          (CONS '(L)
3180                                (LIST
3190                                  (LIST
3200                                    'LIST
3210                                    (LIST 'QUOTE
3220                                          (READLIST
3230                                            (MULTIAPPEND
3240                                                (LIST 'C)
3250                                                PATH
3260                                                (COND (*FLGN* (LIST 'D 'R))
3270                                                      (T (LIST 'R)) ) ) ) )
3280                                      '(CADR L) ) ) ) ) ) NIL )
3290          (LIST SLOTS) )
3300         ((NCONC (SLOT-FUNS-EXTRACT (CAR SLOTS) (CONS 'A PATH))
3310                 (SLOT-FUNS-EXTRACT (CDR SLOTS) (CONS 'D PATH)) )) ) )
3320(DE STRUC-CONS-FORM (STRUC)                                  /*  C:I  -58/06/29-
3330  (COND ((NULL STRUC) NIL)
3340         ((ATOM STRUC) STRUC)
3350         (T (LIST 'CONS
3360                  (STRUC-CONS-FORM (CAR STRUC))
3370                  (STRUC-CONS-FORM (CDR STRUC)) )) ) )
```

Fig. 5 Continued