Ｕｔｉｌｉｓｐを用いた階層データ入出力プログラム

長田　弘康（国鉄・鉄道技術研究所）

１．はじめに

Ｌｉｓｐ言語の特徴として、ｖｅｃｔｏｒなどを除けばオブジェクトをそのまま入力または出力できるのでプログラムの開発やデバックが簡単にすることがある。しかし、一般のユーザがＬｉｓｐのＳ式を扱うことは抵抗があるので、ならかのＳ式を取扱い易い形式に変換したうえで入出力のできるプログラムを作る必要がある。このようなプログラムを個々のエキスパート・システムなどのアプリケーションに対して開発をおこなっていたのではマン・パワーの問題や開発時間に問題がある。以上の問題点を解決するためＳ式のサブセットである階層データの定義を行い、プロトタイプ・システムの為の階層データの入出力を行うプログラムを開発をおこなった。

２．全体の構成

この入出力プログラムでは、各ルーチンを使用する前に階層データ型のあらかじめ行い、その定義を利用して各ルーチンが動作する形式をとる。階層データ型の定義（ＤＥＦＴＹＰＥ）をしたものは、まず定義のチェック（ＴＴＥＳＴ）をおこない、端末からのデータの入力（ＴＲＥＡＤ）、端末へのデータの出力（ＴＰＲｉＮＴ）、修正（ＴＥＤＩＴ）、ファイル入力したＳ式などの検査（ＤＴＥＳＴ）に利用される

端末からの入力はフル・スクリーンで行うのが望ましいが、Ｕｔｉｌｉｓｐの場合は大型計算機で使用されるのでＣＲＴの場合もテレタイプの様に使用することを前提とした。ただし、使い勝手の面から、データを入力した後でＹｅｓ／Ｎｏというような確認入力をすることがなくキイ数を少なくすることを目標とした。

３．階層データの定義

３．１．シンタクス

階層データの定義自身はＳ表現によって表わし、入出力プログラムがこの定義を参照して動作する。以下に文法をＢＮＦ記法によって表わす。

```
＜型＞：：＝　（＜名前＞　＜体＞）
　　｜（＜名前＞　＜体＞　＜暗黙値＞）
＜体＞：：＝　＜簡易体＞｜（＜複合体＞）｜＜リスト体＞
＜複合体＞：：＝　＜簡易体＞　｜＜簡易体＞＜複合体＞
＜簡易体＞：：＝　（ＦＩＸＮＵＭ）
　　｜（ＦＩＸＮＵＭ　＜整数下限＞　ＮＩＬ）
　　｜（ＦＩＸＮＵＭ　ＮＩＬ　＜整数上限＞）
```

| （ＦＩＸＮＵＭ　＜整数下限＞　＜整数上限＞）
| （ＦＬＯＮＵＭ）
| （ＦＬＯＮＵＭ　＜実数下限＞　ＮＩＬ）
| （ＦＬＯＮＵＭ　ＮＩＬ　＜実数上限＞）
| （ＦＬＯＮＵＭ　＜実数下限＞　＜実数上限＞）
| （ＳＹＭＢＯＬ）
| （ＳＹＭＢＯＬ　＜記号列＞）
| （ＳＴＲＩＮＧ）| （ＮＩＬ）| ＜外部名＞

＜記号列＞：：＝　＜記号＞
| ＜記号＞　＜記号列＞

＜リスト体＞：：＝（ＬＩＳＴ　＜リスト型＞）

＜リスト型＞：：＝＜直列リスト型＞| ＜直列リスト型＞（＜直列リスト型＞）
| （＜直列リスト型＞）

＜直列リスト型＞：：＝＜型＞| ＜型＞　＜直列リスト型＞

＜名前＞、＜外部名＞，＜記号＞　～　Ｌｉｓｐのシンボル
＜整数上限＞、＜整数下限＞　～　Ｌｉｓｐの整数
＜実数上限＞、＜実数下限＞　～　Ｌｉｓｐの浮動小数点数

上記で『～』は左辺が右辺と同等な文法によるという意味で用いている。

３．２．セマンテックス

データ型の定義は、Ｓ式のサブ・セットに対応しており、データ型を用て入出力されるのはＬｉｓｐのオブジェクトである。そこて、データ型の定義に対応するオブジェクトについて説明する。

まず、シンタクスで用いた用語の説明をする。

（１）　＜型＞はデータ型全体を表わす。

（２）　＜名前＞はデータ型を識別するのに用いる。

（３）　＜体＞はデータ型本体を表わす。データの階層化の方法、データの種別および、データの有効範囲を与える。後で詳しく述べる。

（４）　＜暗黙値＞は、＜体＞が＜簡易体＞または＜複合体＞のときのみ有効であり、＜体＞の仕様を満足するデータを置く。＜暗黙値＞は、データの入力時の手間を少なくする為に用いる。

次に、＜体＞に対応するＬｉｓｐのオブジェクトについて述べる。

（ａ）　簡易体

（ａ．１）　（ＦＩＸＮＵＭ）：整数に対応する。

（ａ．２）　（ＦＬＯＮＵＭ）：浮動小数点数に対応する。

（ａ．３）　（ＳＹＭＢＯＬ）：シンボルに対応する。

（ａ．４）　（ＳＴＲＩＮＧ）：ストリングに対応する。

（ａ．５）　（ＦＩＸＮＵＭ　＜整数下限＞　ＮＩＬ）：整数下限以上の整数が対応する。

（ａ．６）　（ＦＩＸＮＵＭ　ＮＩＬ　＜整数上限＞）：整数上限以下の整数が対応する。

（ａ．７）　（ＦＩＸＮＵＭ　＜整数下限＞　＜整数上限＞）：整数下限以上整数上限以下の整数が対応する。

（ａ．８）　（ＦＬＯＮＵＭ　＜実数下限＞　ＮＩＬ）：実数下限以上の浮動小数点数が対応する。

（ａ．９）　（ＦＬＯＮＵＭ　ＮＩＬ　＜実数上限＞）：実数上限以下の浮動小数点数が対応する。

（ａ．１０）（ＦＬＯＮＵＭ　＜実数下限＞　＜実数上限＞）：実数下限以上実数上限以下の浮動小数点数が対応する。

（ａ．１１）（ＳＹＭＢＯＬ　＜記号１＞　＜記号２＞　．．．＜記号ｎ＞）シンボル＜記号１＞，＜記号２＞，．．．＜記号ｎ＞のいずれかのシンボルが対応する。

（ａ．１２）（ＮＩＬ）：オブジェクトが対応する。

（ａ．１３）＜外部名＞：＜外部名＞と同じ名前を持つデータ型の体が引用される。

（ｂ）　複合体

＜体＞が（＜簡易体１＞　＜簡易体２＞　．．．　＜簡易体ｎ＞）の場合には、＜簡易体１＞から＜簡易体ｎ＞のいずれかに対応するオブジェクトが対応する。

（ｃ）　リスト体

（ｃ．１）　（ＬＩＳＴ　＜型１＞　＜型２＞　．．．　＜型ｍ＞）：＜型１＞で定義されるオブジェクトから＜型ｍ＞で定義されるオブジェクトを要素とするリストが対応する。

（ｃ．２）　（ＬＩＳＴ　＜型１＞　＜型２＞　．．．　＜型ｍ＞（＜型１＞　＜型２＞　．．．　＜型ｎ＞））：＜型ｉ＞で定義されるオブジェクトをｅｉと書き、＜型ｊ＞により定義されるオブジェクトをｅｊと書くと、ｅ１からｅｍと０個以上のｅ１からｅｎの繰り返しからなるリストが対応する。

（ｃ．３）　（ＬＩＳＴ　（＜型１＞　＜型２＞　．．．　＜型ｎ＞））：＜型１＞で定義されるオブジェクトから＜型ｎ＞で定義されるオブジェクトの１個以上の繰り返しからなるリストが対応する。

入出力プログラムの関係上、データ型の定義に以下の制約条件がある。

（ア）再帰的な構造はできない。つまり、外部名を用いて自分自身を引用することはできない。

（イ）リスト体（外部名によって引用される体を含めて）は、暗黙値を取ることはできない。

（ウ）複合体では、リスト体（外部名によって引用される体を含めて）と複合を取ることはできない。

（エ）リスト体の要素では暗黙値を持つ要素以後では必ず各要素は暗黙値を持たなければならない。

データ定義の例：　年月日は以下のように定義される。

```
（ＤＡＴＥ　（ＬＩＳＴ　（ＹＥＡＲ　（ＦＩＸＮＵＭ））
                （ＭＯＮＴＨ　（ＦＩＸＮＵＭ　１　１２））
```

（ＤＡＹ　（ＦＩＸＮＵＭ　１　３１））））

４．データ型の定義の方法

データ型を入出力プログラムで利用する為には、あらかじめデータ構造を貯える必要がある。

ＤＥＦＴＹＰＥ　｛Ｍａｃｒｏ｝　ｎａｍｅ　ｂｏｄｙ　．　ｄｅｆａｕｌｔ

名前が＜ｎａｍｅ＞、体が＜ｂｏｄｙ＞、暗黙値が＜ｄｅｆａｕｌｔ＞とするデータ型を定義する。

ＣＬＥＡＲＴＹＰＥ

データ型の定義を全て空にする。

ＥＲＡＳＥＴＹＰＥ　ｎａｍｅ

名前が＜ｎａｍｅ＞のデータ型を消去する。

以上の様なプログラムが準備されている。データ型は特定のシンボルの値としてペア・リストの形で貯えられているのにすぎない。データ型の数が増えた場合にはハッシュ化などの必要があるだろうが現状では今の方法で十分である。

なお、データ型のシンッタクス及び制約条件のチェックは以下のプログラムにより行い個々の入出力ルーチンではチェックはおこなわない。

ＴＴＥＳＴ　｛Ｍａｃｒｏ｝　ｎａｍｅ

５．会話的入力プログラム

定義されたデータ型を参照して端末から会話的にデータを入力するプログラムであり、データ型を参照することにより、その構造を満たすデータのみが入力される。このチェック機構があるためにアプリケーションでは処理のみを集中的に作成すればよいことになる。

ＴＲＥＡＤ　ｎａｍｅ

名前＜ｎａｍｅ＞に対応するデータ型に従ってデータの入力を行い、そのデータを値として返す。

データ型の定義から入力時のプロンプトなどが生成されるがそれだけではデータの意味がわからなくなる場合があるので、簡易体については『？』を入力することにより、その簡易体に付随したメセージが出力される機構を準備した。

ＤＥＦＣＯＭＭＥＮＴ　｛Ｍａｃｒｏ｝　ｎａｍｅ　．　ｃｏｍｌｉｓｔｓ

データ型の名前＜ｎａｍｅ＞に対して、コメント・メッセージ＜ｃｏｍｌｉｓｔ＞を与える。コメント・メッセージはストリングでなければならない。

前に示した年月日のデータ型を例にとりＴＲＥＡＤの操作例を示す。

```
・データ型の定義
〉(DEFTYPE DATE (LIST
〉                (YEAR (FIXNUM))
〉                (MONTH (FINUM 1 12))
〉                (DAY (FIXNUM 1 31))))
((DATE (LIST ???)))
・コメント・ストリングの定義
〉(DEFCOMMENT MONTH "1 --- Jan"
〉                   "2 --- Feb"
〉                   ".........")
("1 --- Jan" "2 --- Feb" ".........")
・ＴＲＥＡＤによるデータ入力
〉(TREAD 'DATE)
DATE ::I::
.YEAR:〈〈FIXNUM〉〉==I=〉 1984
.MONTH:〈〈1--12〉〉==I=〉 Feb
.MONTH:〈〈1--12〉〉?=I=〉 ?
     1 --- Jan
     2 --- Feb
     .........
.MONTH:〈〈1--12〉〉?=I=〉 3
.DAY:〈〈1--31〉〉==I=〉 〈
.MONTH:〈〈1--12〉〉|3|==R=〉 2
.DAY:〈〈1--31〉〉==I=〉 14
(1984 2 14)
```

図１　ＴＲＥＡＤの実行例

図１に示す様にＴＲＥＡＤプログラムでは、データ型の内容を参照してプロンプトの生成及び入力されたデータのチェックを行っている。また、ＴＲＥＡＤでは、単にデータの入力をするだけではなく『＜』などの文字によりデータの打ちまちがいを修正する機能をもつ。『＜』などの特殊文字の機能としては以下のようなものが使用できる。

（１）暗黙値呼出し　『井』：暗黙値が定義されている場合、『井』を入力する　と暗黙値を入力したのと同じ効果がえられる。

（２）逆もどし　『＜』：入力点を一つ前に戻す。

（３）繰り返しの切除　『！』：リスト型の繰り返し部分を切除する。

（４）コメント・メッセージの表示　『？』

（５）リスト型の入力終了　『）』：繰り返し部分や固定部分の途中でそれ以後の要素に暗黙値がある場合にはその値が補充される。また、繰り返し部分の初めの場合には当該リスト型の入力の終了を意味する。

（６）入力の終了　『¥』：『＜』で入力位置を変更したときに用いる。

注：リスト型の繰り返しの追加の機能は特殊文字ではなくて『＜』により繰り返し部分の先頭に戻ると自動的に追加モードになる。繰り返しの追加をスキップするには単に空行を入力すればよい。

入力の手間を省く為に空行の入力により『＜』により既にデータの入力せれている所では前のデータが、暗黙値の定義がある要素ではその暗黙値を入力するのと同じ意味を持つ。つまり、空行が暗黙動作に対応している。また、１行内の多くのデータ要素の入力は特殊文字を含めて空白で区切ることにより可能である。（これは、ＴＲＥＡＤルーチンにより実現した機能ではなくＵｔｉｌｉｓｐの入力機構による。）

ＴＲＥＡＤプログラムは、データに関するスタックとデータ型に関するスタックの両方を持ち、入力された内容に従って２つのスタックを操作するプログラム

である。面倒なのは、リスト体をもつデータ型のときの繰り返しに関するスタック操作である。

6．製書出力プログラム

データ型の定義に合ったデータを端末に出力するプログラムを用意している。

TPRINT name data

名前<name>に対応するデータ型に従ってデータ<data>の出力を行う。

```
) (TPRINT 'DATE '(10984 2 14))
DATE
.YEAR:1984    .MONTH:2    .DAY:14
```

図２　TPRINTの実行例

データの出力で問題となるのはリスト体をデータ型が持つ場合である。TPRINTでは、リスト型の名前及びレベルや繰り返しの数などの付帯情報を印字したのちに改行をしてリスト体に対応するデータの内容を印字するのが基本的ルールである。リスト体の内容に印字については、もし全ての要素が基本体を持つならば、１行になるべく多くの要素について印字をし、要素のなかに基本体以外の体を持つ型がある場合には各要素の印字ごとに改行をする。

7．会話的修正プログラム

データを端末から会話的に修正する為のエデッタを準備した、データ型を参照することにより、その構造を満たすようにデータを修正する。このチェック機構があるためユーザの作成するプログラムの負荷が低減される。

TREREAD name data

名前<name>を持つデータ型に従ってデータ<data>を再入力して、変更されたデータを値として返す。

TREREADプログラムはTREADルーチンを変更モードにすることにより実現しているので、操作の方法についてはTREADルーチンと同じである。

TEDIT name data

名前<name>に対応するデータ型に従ってデータ<data>を会話的に修正を行い、の修正されたデータを値として返す。

TEDITルーチンは、コマンド式のエデッタであり、コマンドはLispのシンボルからなる修正を行うためのものと，整数から成る対象データを詳細化するためのものがある。

整数コマンドは、現在の修正対象がリスト体に対応しているときに有効であり、整数コマンドで示される番号の要素に修正の対象を移動する。要素と整数の対応は、コマンド『P』または『？』により情報を出力させる。

シンボル・コマンドは修正の対象となっているデータ型と上位のデータ型により有効であったり、無効であったりするが、コマンド入力時のプロンプトに有効なコマンドが全て表示されるので、ユーザは、有効なコマンドがなんであるのか

```
) (SETQ DATE-1 '(1984 2 14))
(1984 2 14)
) (TEDIT 'DATE DATE-1)
DATE:in EDIT mode
COMMAND=(?/P/PP/Q)===) 2
.MONTH:in EDIT mode
COMMAND=(?/(/)/¥/S/L/U/P/PP/R/Q)===) R
.MONTH:((1--12)) [2]=R=) 5
.MONTH:in EDIT mode

?:"Help"                          (:"Move Before Element"
):"Move Next Element"             ¥:"Go to Toplevel"
S:"Move Start Element"            L:"Move Last Element"
U:"Scope Level up"                P:"Print Now Element"
PP:"Precise Print Now Element"    R:"Replace Now Element"
Q:"Quit Edit"
SCOPE DATA=5

.MONTH:in EDIT mode
COMMAND=(?/(/)/¥/S/L/U/PP/Q)===) L
.DAY: in EDIT mode
COMMAND=(?/(/¥/S/U/P/PP/R/Q)===) R
.DAY:((1--31))[14]==R=) 7
.DAY: in EDIT mode
COMMAND=(?/(/¥/S/U/P/PP/R/Q)===) ¥
DATE: in EDIT mode
COMMAND=(?/P/PP/R/Q)===) PP
DATE:
.YEAR:2  MONTH:5  DAY:7
DATE in EDIT mode
COMMAND=(?/P/PP/R/Q)===) Q
(1984 5 7)
```

図３　ＴＥＤＩＴの使用例

を知ることができる。以下にシンボル・コマンドの種類と機能をしめす。

（１）『？』：詳細な情報の表示　　（２）『＜』：前の要素への移動
（３）『＞』：次の要素への移動　　（４）『！』：消去
（５）『¥』：最上位レベルへの移動　　（６）『Ｓ』：初めの要素への移動
（７）『Ｌ』：最後の要素への移動　　（８）『Ｕ』：レベルを１つあげる
（９）『Ｐ』：表示　　（１０）『ＰＰ』：詳細な表示
（１１）『Ｒ』：修正　　（１２）『Ｉ』：入力
（１３）『Ａ』：追加　　（１４）『Ｑ』：終了

現在のＴＥＤＩＴルーチンでは、コマンドの取消し（ｕｎｄｏ）や要素の複写などが無いという問題点がある。

８．データ検査プログラム

ファイルなどから入力したデータを、データ型を参照することにより、その構造を満たすかを検査し、誤りがある場合にはエラー・メーセッジを出力し、修正可能なものは、データを変更する。

ＤＴＥＳＴ　ｎａｍｅ　ｄａｔａ

名前＜ｎａｍｅ＞を持つデータ型を参照して、データ＜ｄａｔａ＞の検査を行い、修正されたデータを返す。

データの中に特殊文字『＃』があるときには、データ型に定義された暗黙値を書くのと同様な動作をし、またリスト型において対応するデータの数が少ない場合は、それ以降のデータ型に暗黙値が定義されているときはデータに暗黙値が補充される。

データ検査ルーチンでは、対象となるデータとデータ型をマッチングするのが主な内容である。マッチングをする場合、データ型が基本体をもつ場合には簡単に表１のようにデータ型との対応できまる評価値を用いて行う。評価値が４の場合にはデータが受理される。評価値が３の場合にはデータは受理できないが、類似のデータに変換をする（表１では変換データとして表わしている）。評価値が２以下の場合にはデータが変換できない場合でありエラーとなる。

データ型がリスト体をもつ場合には、データがアトムの場合にはエラーになる。データもリストの場合にはリストの要素とリスト型の要素となる型で局所的に処理を次のように行う。

（１）処理の対象のなるリストの要素とリスト型の要素の評価値が４の場合にはその要素を受理して次の要素を処理する。

（２）それ以外の場合には、まず次の３つの評価値を計算する。

V_{21}　＝現在の処理対象となっているデータと現在のデータ型の評価値。

V_{12}　＝現在の処理対象となっているデータと次のデータ型の評価値。

V_{22}　＝次の処理対象となるデータと次のデータ型の評価値。

（２．１）評価値の最大値がV_{12}であり、それが３以上の場合には、現在のデータがたらないものとして＄ＥＲＲＯＲ＄を受理したものとする。次の処理対象をデータに対しては変化させずに、データ型については１つ前に進める。

（２．２）評価値の最大値がV_{21}であり、それが３以上の場合には現在の処理対象になっているデータが余分にあるものとして、次の処理対象をデータに対しては変化させずに、データ型については１つ前に進める。

以上がリストに対する基本的な処理の方法であるが、他に暗黙値の埋め込みの処理などがある。このアルゴリズムは局所的に処理を行うので、リスト体を構成する型に変化が多くないと誤りの含んだデータに対して必要以上のエラーが検出されてしまうことになる。これを回避する方法としてはリスト全要素についての大域的な判断をすることなどが考えられる。

表1　ＤＴＥＳＴで用いるマッチングの評価値

データ型の体	データ	評価値	変更データ
(ＮＩＬ)	任意	4	― ― ―
(ＦＩＸＮＵＭ)	整数 浮動少数点数 任意	4 3 1	― ― ― 整数にしたもの ― ― ―
(ＦＬＯＮＵＭ)	浮動少数点数 整数 任意	4 3 1	― ― ― 浮動少数点数にしたもの ― ― ―
(ＳＹＭＢＯＬ)	シンボル 任意	4 1	― ― ― ― ― ―
(ＳＴＲＩＮＧ)	ストリング シンボル 任意	4 3 1	― ― ― ストリングにしたもの ― ― ―
(ＦＩＸＮＵＭ ｉｍｉｎ ＮＩＬ)	整数＞＝ｉｍｉｎ 浮動少数点数＞＝ｉｍｉｎ 数値 任意	4 3 2 1	― ― ― 整数にしたもの 整数にしたもの ― ― ―
(ＦＩＸＮＵＭ ＮＩＬ ｉｍａｘ)	整数＜＝ｉｍａｘ 浮動少数点数＜＝ｉｍａｘ 数値 任意	4 3 2 1	― ― ― 整数にしたもの 整数にしたもの ― ― ―
(ＦＩＸＮＵＭ ｉｍｉｎ ｉｍａｘ)	ｉｍｉｎ＜＝整数＜＝ｉｍａｘ ｉｍｉｎ＜＝浮動少数点数＜＝ｉｍａｘ 数値 任意	4 3 2 1	― ― ― 整数にしたもの 整数にしたもの ― ― ―
(ＦＬＯＮＵＭ ａｍｉｎ ＮＩＬ)	ａｍｉｎ＜＝浮動少数点数 ａｍｉｎ＜＝整数 数値 任意	4 3 2 1	― ― ― 浮動少数点数にしたもの 浮動少数点数にしたもの ― ― ―
(ＦＬＯＮＵＭ ＮＩＬ ａｍａｘ)	浮動少数点数＜＝ａｍａｘ 整数＜＝ａｍａｘ 数値 任意	4 3 2 1	― ― ― 浮動少数点数にしたもの 浮動少数点数にしたもの ― ― ―
(ＳＹＭＢＯＬ ． ｓｙｍｂｏｌｓ)	シンボル∈ｓｉｎｂｏｌｓ シンボル 任意	4 2 1	― ― ― ― ― ― ― ― ―

９．あとがき

エキスパート・システムなどのＬｉｓｐの応用プログラムを効率良く作成する為に、入出力ライブラリ・プログラムを作成した。本ライブラリを使用すること

により入出力はデータ仕様を書くだけですむことになる。

本入出力ライブラリ・プログラムについでの改善すべき点について述べる・

（１）データ型の種類が少ない。特にプロダクション・システムのルールの様な構文をもったデータは扱いにくい。

（２）入力が行単位なので、使い勝手が悪い。今後はフル・スクリーン又はビット・マップデスプレイでの操作が可能なものを考えていく予定である。

（３）複合的なデータの検査ができない。つまり、『月日』のようなデータ型について、『月』が１から１２ということは検査できるが、４月３１日というデータをリジェクトする機能はない。現状では、本入出力ライブラリの外部でチェックする以外に方法はない。この複合的なデータ検査機能は望ましいが（２）の問題を解決しないと入力時にエラーをプログラムが発見した場合にどこを修正するかを指定しなくてはならない。『月日』のように単純ではなくて検査が複数の要素にわたる場合は更に困難な状況がありうる。

１０．文献

（１）Ｃｈｉｋａｙａｍａ，Ｔ．：″Ｕｔｉｌｉｓｐ　Ｍａｎｕａｌ″，ＭＥＴＲ　８１－６，Ｄｅｐａｒｔｍｅｎｔ　ｏｆ　ｍａｔｈｅｍａｔｈｃａｌ　ｅｎｇｉｎｅｅｒｉｎｇ，　Ｕｎｉｖｅｒｓｉｔｙ　ｏｆ　Ｔｏｋｙｏ，１９８１

（２）長沢孝次、長田弘康、後藤和則：″有機物質の赤外線吸収スペクラム解析の知識工学的手法の応用″、情報処理学会２８回大会、１０５９－１０６０（１９８４）

（３）長田弘康、長沢孝次、後藤和則：″赤外線吸収スペクトラム解析プログラムの構成方法″，情報処理学会２８回大会、１０６１－１０６２　（１９８４）

（４）長田弘康：″Ｌｉｓｐを使った入出力プログラム（その１）″、鉄道技術研究所速報Ａ－８４－１２２（１９８４）