ICOT Technical Memorandum: TM-0784

TM-0784

SEMACSのインストール方法

東　吉郎(JIPDEC)

August, 1989

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

ＳＥＭＡＣＳの
インストール方法

ＳＥＭＡＣＳのインストール方法

1．インストール方法

①ライブラリアンを生成する。

②ファイル・マニピュレータを生成する。

③ＳＥＭＡＣＳフロッピーをユニット０に挿入する.

④リンク名＞ｆｄ０でマウントする。

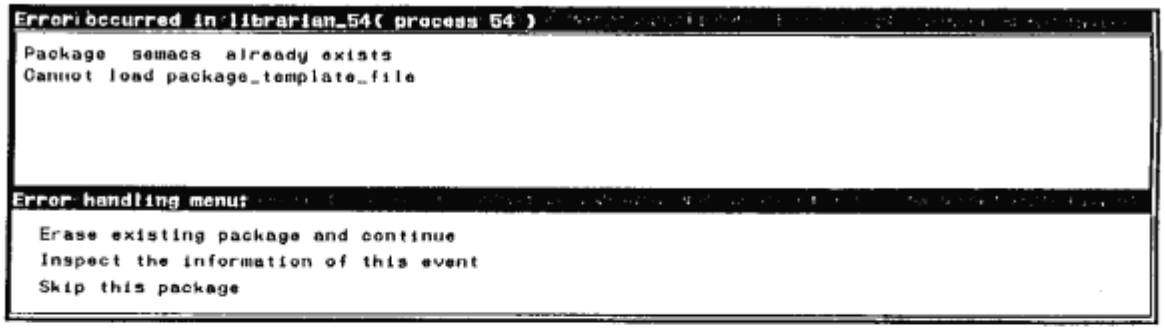
⑤ライブラリアンのコマンドＥｘｅｃｕｔｅを選択する。

⑥Command>[load _package _template(semacs,　">fd0>semacs.ptf　")]と入力し、
キャリジ・リターンを行う。

⑦約３分後インストールが終了する。

注意）

既にパッケージＳＥＭＡＣＳが存在していると、次のシチュエーション・メッセージが表示されるので、Erase existing package and continue （既存のパッケージを削除し、新規にパッケージを生成する）を選択する。すると確認のメニューが表示されるので、ERASE を選択する。



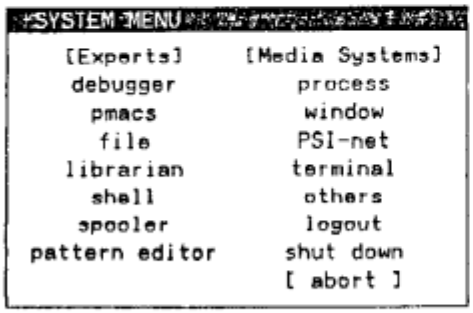
2．インストール後

インストール終了後、パッケージsemacsが存在していて、パッケージ環境simposが
このパッケージを継承しているかを確認する。

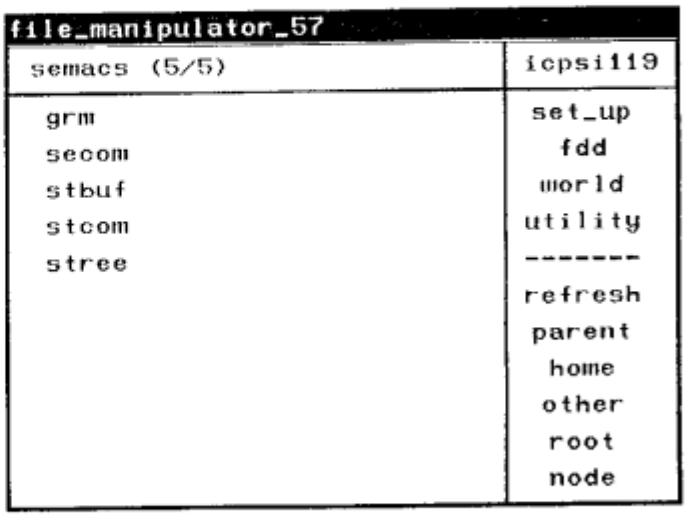
3.

1)．　(図1) のｆｉｌｅを選ぶ。

(図1)

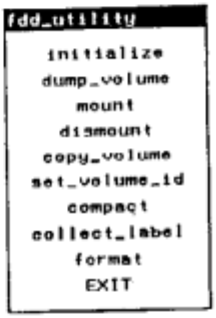


(図2)



2)．　(図2) のｆｄｄを選ぶ。

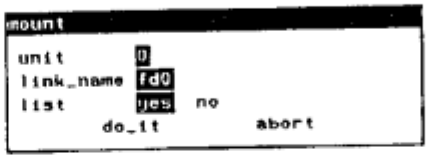
(図3)



3)．　" GRAMMAR" のＦＤをＦＤ０に入れ (図3) のｍｏｕｎｔを選ぶ。

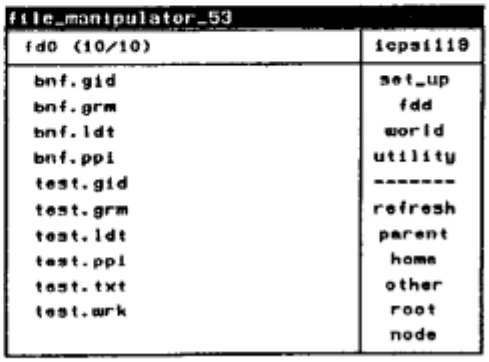
(図4)

4)．　(図4) のｄｏ　ｉｔを選ぶ。



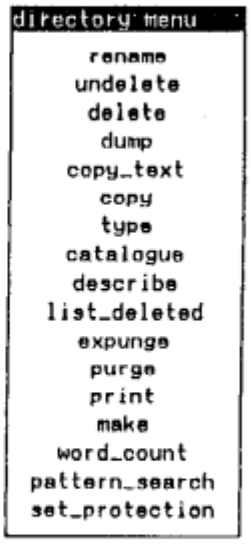
(図5)

5)．　(図5) のｆｄ０ (１０／１０) を選ぶ。



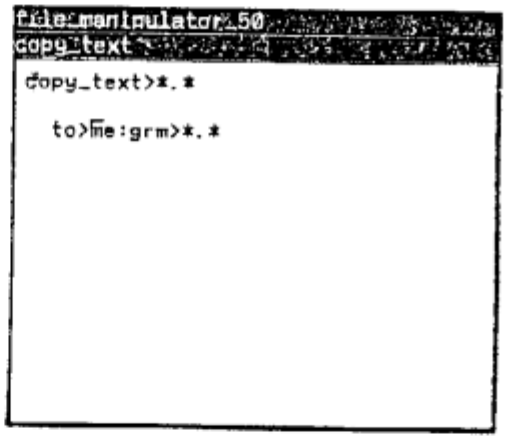
6）．　（図6）のｃｏｐｙ＿ｔｅｘｔを選ぶ。

（図6）



7）．　（図7）の" ｃｏｐｙ＿ｔｅｘｔ＞" に対して＊．＊＜ＣＲ＞を入力

（図7）



8）．　（図7）の" ｔｏ＞" に対してｍｅ：ｇｒｍ．＊．＊＜ＣＲ＞を入力。

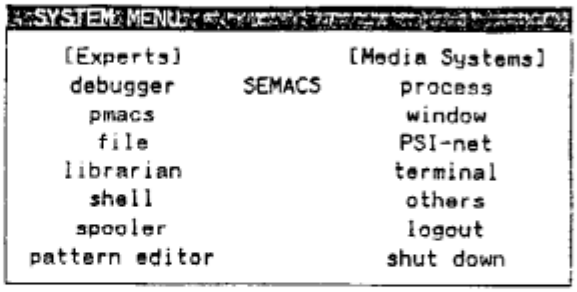
9）．　ログアウトする。

4.

1) ユーザ名ｓｕｐｅｒｕｓｅｒにログインする。

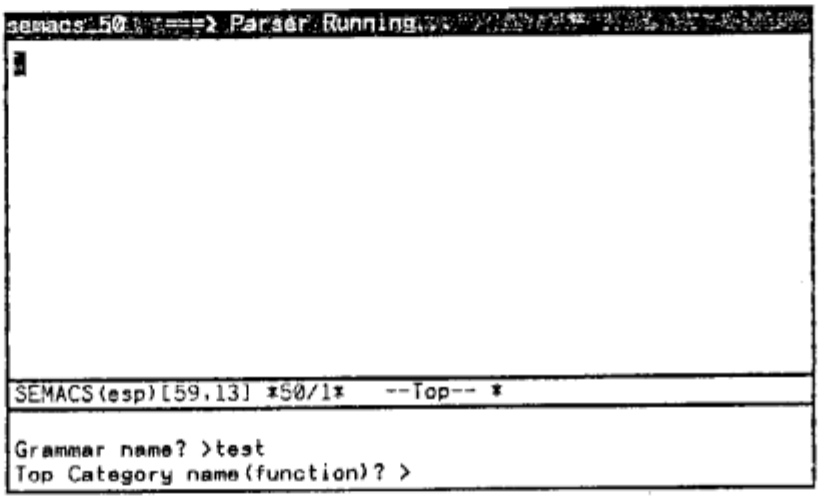
2) (図8)のＳＥＭＡＣＳを選ぶ。

(図8)



3) (図9)にａ(ｂ，ｃ)と入力する。

(図9)



4) (図9)にｃ－ｘのｃ－ｐと操作する。

5) "Ｇｒａｍｍａｒ　ｎａｍｅ？　>"に対してｔｅｓｔ<ＣＲ>を入力。

6) "Ｔｏｐ　Ｃａｔｅｇｏｒｙ　ｎａｍｅ(ｆｕｎｃｔｉｏｎ)？>"に対して<ＣＲ>を入力。

7) 抜ける時は、Ｔ続いてｃ－ｘのｃ－ｚでｐｍａｃｓを抜ける。

8) 完了。

SEMACS使用説明書

（第二版）

平成元年７月１日

財団法人　新世代コンピュータ技術開発機構

本説明書は逐次型推論マシンＰＳＩ（Personal Sequential Inference Machine）上の構造エディタＳＥＭＡＣＳ（Structure Extended pMACS）の使用法について説明したものである。

ＳＥＭＡＣＳ使用説明書　目　　次

１．概説 …… 1
　１．１　構造エディタを使用するメリット …… 1
　１．２　用語の説明 …… 4
　１．３　ＳＥＭＡＣＳの使用法の概略 …… 6
　１．４　実行環境 …… 7

２．インストール方法 …… 8
　２．１　供給ＦＤＤ …… 8
　２．２　ローディング …… 9
　２．３　システムメニューへの登録 …… １０

３．文法の定義法 …… １１
　３．１　構文要素 …… １２
　３．２　文法の記述法 …… １３
　３．３　文法定義ファイルの作成法 …… １８

４．構造の編集法 …… １９
　４．１　起動法と画面の説明 …… １９
　４．２　ストリングモード …… ２０
　４．３　エリアの移動 …… ２３
　４．４　エリアの削除、復元 …… ２６
　４．５　構造の探索、置換 …… ２８
　４．６　リスト要素の処理 …… ３０
　４．７　ホロフラスティング …… ３１
　４．８　文法案内 …… ３２
　４．９　バッファの切換え …… ３４
　４．10　セーブ、ロード …… ３６

５．木表示機能 …… ３７
　５．１　エリアの木表示 …… ３７
　５．２　ノードの名前表示 …… ４０
　５．３　ビュー・ウインドウのスクロール …… ４１
　５．４　ユーザが選択したノードをルート・ノードする部分木の表示 …… ４３
　５．５　文法案内 …… ４４

付録　構造モードでのコマンド一覧

1．概説

1．1　構造エディタを使用するメリット

本節ではＳＥＭＡＣＳに限らず構造エディター般の利点について説明する。

図１－１に示す文法とそれをもとに作られた図１－２の構造を例に説明する。
なお、図１－１に示す文法および図１－２の構造を木の形で表示すると、それぞれ図１－３、図１－４のようになる。

```
辞書項目
    見出し
    品詞
    語形変化
        活用行
        活用型
    参考語
        同義語
        反義語
```

図１－１　辞書の一項目の文法の例

```
生きる
動詞
    カ行
    上一段
    生活する
    死ぬ
```

図１－２　図１－１をもとにして作られた構造の例

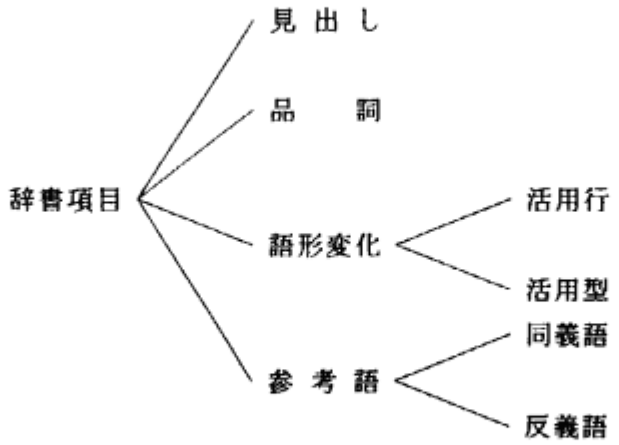


図１－３　図１－１の文法の木表示

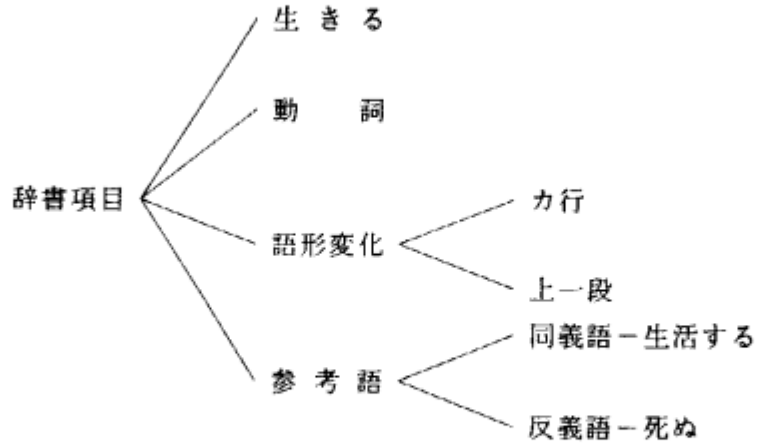


図１－４　図１－２の構造の木表示

エディタが文法を「知っている」ことにより、従来のテキスト・エディタではできなかった（１）～（９）のことができる。

（１）データベースのスキーマのような木構造で、部分木の置換や削除など、構造単位の編集ができる。

（２）ホロフラスティングにより全体的な構造が一目でつかめる。

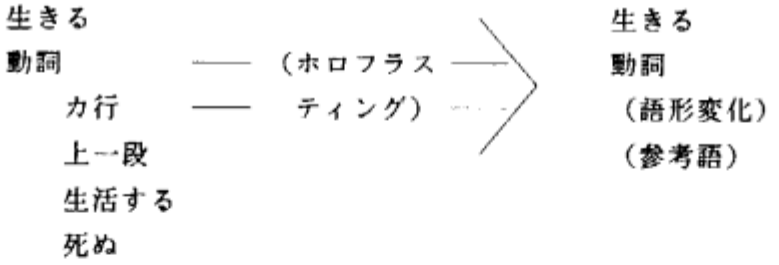


図１－５　ホロフラスティングの例

（３）文法案内ができる。

ユーザはその文法が段階的かつ詳細に示されるのにしたがって、データなどを入力していけばよい。この機能はユーザが初心者の場合や、構造の詳細を忘れてしまった場合などに便利である。

（４）プログラム開発時などに、ユーザはより本質的な部分に集中できる。

たとえばPascalのようなプログラムを編集する場合、ifやwhile のような構造上のキーワードは構造エディタにより、構文解析時にチェックされる。したがってユーザはタイプミスやその他インデンテーションなどに気を使うことなく、プログラミングそのものに集中できる。

（5）ドキュメントの標準化ができる。

仕様書などを構造エディタで作成すれば、章立てやインデンテーションその他の構成を統一することができる。

（6）統一したプログラミング環境を構築できる。

構造エディタ内部で扱っているデータの構造表現を利用すれば、エディタに関連する他のプログラミング・ツールも容易にかつ統一した形で構築できる。

（7）構造的な正しさが保証される。

すでに述べたように構造上のキーワードは構造エディタにより構文解析時にチェックされるので、たとえばend のつけ落としのようなことがなくなる。

１．２　用語の説明

本説明書で使用する用語について説明する。

文法

辞書項目は見出し、品詞、語形変化、参考語から成り、さらに語形変化は活用行、活用型、参考語、同義語、反義語からなるというように、ある情報とある情報との関係を定めるもの。

構造

文法にしたがってつくられた実際のデータ。

終端記号

構造の最少単位。それ以上小さい構造に分割できないもの。図１－４の例では「生きる」、「動詞」、「カ行」、「上一段」、「同義語」、「反義語」。

非終端記号

それ以上小さい構造に分割可能な構造。図１－４の例では「辞書項目」、「語形変化」、「参考語」。

エリア

現在編集対象となっている構造の単位。

ボックス

ＳＥＭＡＣＳのウインドウ上で、エリアを白黒反転して表示している領域。

ストリングモード、構造モード

ＳＥＭＡＣＳには２つのモード、ストリングモードと構造モードとがある。ストリングモードは文字列を入力、編集するモードである。ＳＥＭＡＣＳがストリングモードの場合ＳＩＭＰＯＳのＰＭＡＣＳと同じである。

構造モードは構造単位の編集を行うモードであり、構造エディタならではの操作が行われる。ただし、構造モードでは文字列の入力、編集はできない。

top_category（トップカテゴリ）

その文法の最上位の概念、情報。図１－１の例では辞書項目（図１－３で辞書項目がルートノードになっていることに注意）。

area_top _category（エリアトップカテゴリ）

その時エリアとなっている構造に対応する、文法の部分の最上位の概念、情報。図１－２の例で、「カ行」、「上一段」の両方がエリアになっている場合（図１－６）、area_top _categoryは語形変化（図１－４を参照しながら考えるとわかりやすい）。

図１－６　エリアを矩形枠で表示した場合

１．３　ＳＥＭＡＣＳの使用法の概略

図１－７に示す。本書の構成もこれに沿ったものになっている。

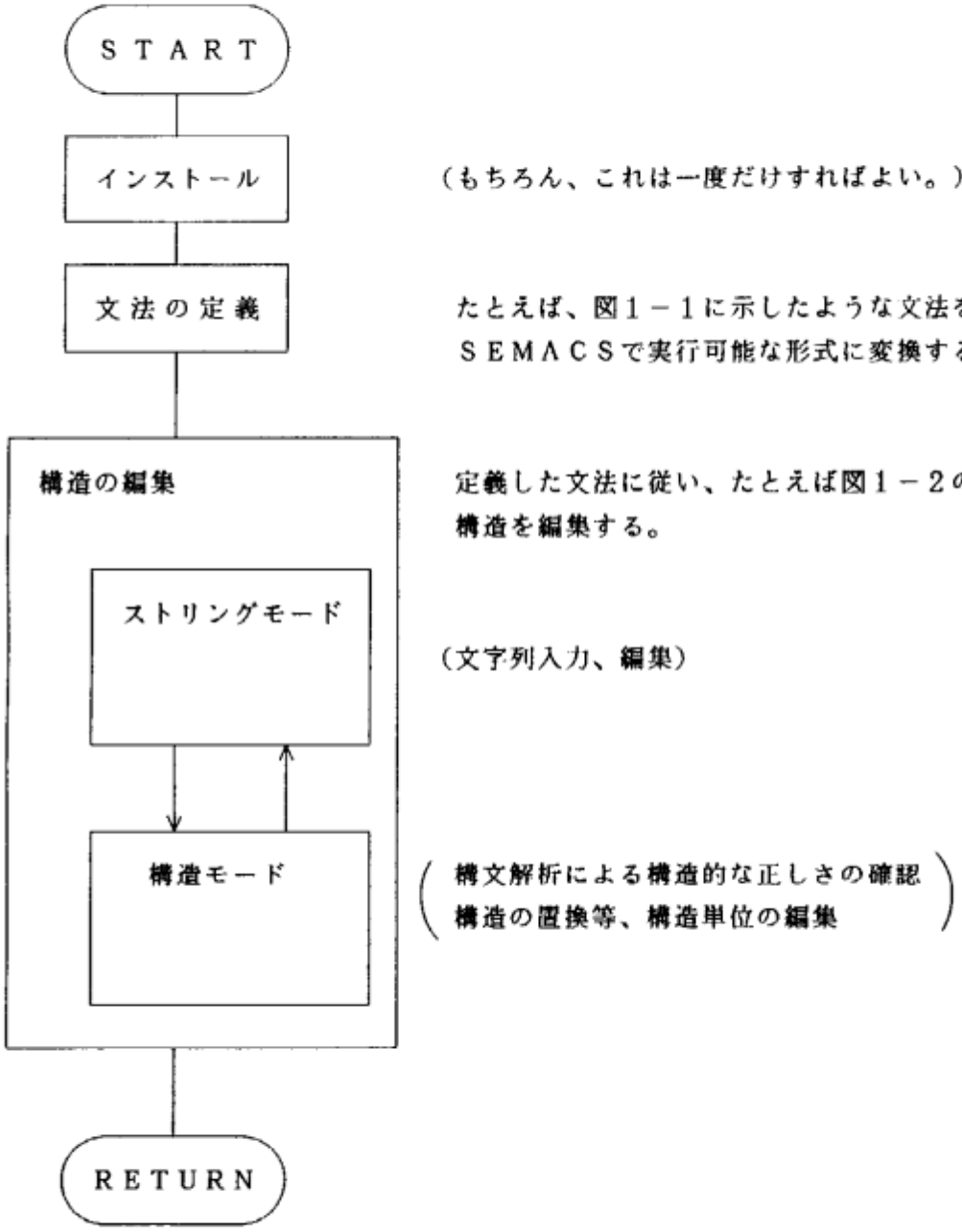


図１－７　ＳＥＭＡＣＳの使用法の概略

１．４　実行環境

ＳＥＭＡＣＳはＳＩＭＰＯＳ５．１版上で動く。詳しくは「２　インストール法」参照のこと。

内部的には表示および文字列編集部分にＳＩＭＰＯＳの対応する版のＰＭＡＣＳを流用し、そのうしろに構造を操作、編集する部分がついたかたちになっている（図１－８参照）。

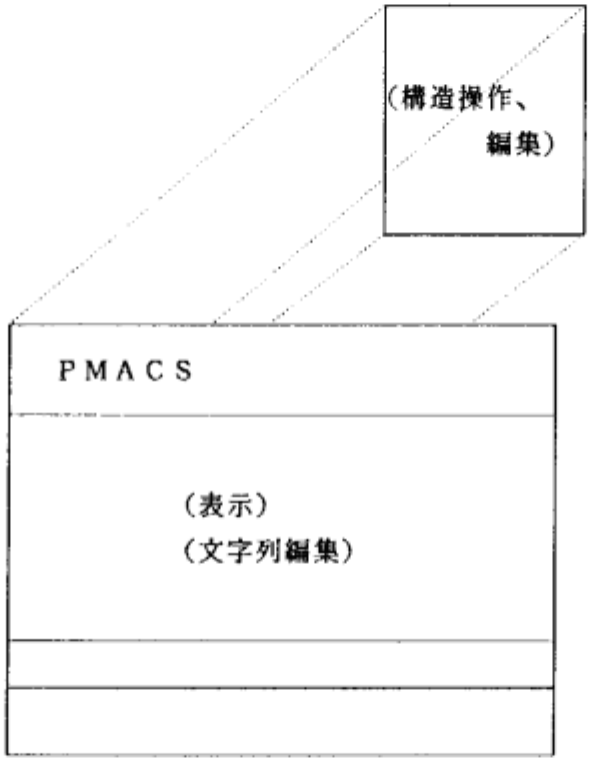


図１－８　ＳＥＭＡＣＳの構成

2．インストール方法

2．1　供給ＦＤＤ

ＳＥＭＡＣＳは以下に示す２枚のＦＤＤで供給する。

表２－１　供給ＦＤＤ一覧

ＮＯ	ラベル名	内　　容
1	ＳＥＭＡＣＳ	構造エディターパッケージテンプレートファイル
2	ＧＲＡＭＭＡＲ	構造エディター文法ファイル

２．２　ローディング

（１）パッケージテンプレートファイル（SEMACS）のロード

ライブラリアンを立上げてパッケージテンプレートファイルを一括ロードする。（詳細はＳＩＭＰＯＳマニュアル参照）

（２）文法定義ファイルのロード

文法定義ファイル用のディレクトリを論理名ｇｒｍ：　で作成する。（ディレクトリ名は任意）ｇｒｍ：　の下にＧＲＡＭＭＡＲ中のすべてのファイルをコピーする。コピーされるファイル名を以下に示す。

ｂｎｆ．ｇｉｄ
ｂｎｆ．ｇｒｍ
ｂｎｆ．ｌｄｔ
ｂｎｆ．ｐｐｉ
ｔｅｓｔ．ｇｉｄ
ｔｅｓｔ．ｇｒｍ
ｔｅｓｔ．ｌｄｔ
ｔｅｓｔ．ｐｐｉ

なお、文法定義ファイルはユーザの参考のために配布する。詳しくは３．３文法定義ファイルの作成法を参照。

２．３　システムメニューへの登録

ＳＥＭＡＣＳをシステムメニューへ登録する。登録方法の詳細は、ＳＩＭＰＯＳのマニュアルを参照する事。なお、登録するＳＥＭＡＣＳのパッケージ及びクラス名は以下のとおりである。

（１）パッケージ名 …… ｓｅｍａｃｓ

（２）クラス名 …… ｓｅｍａｃｓ

以降、システムメニューよりＳＥＭＡＣＳが使用可能となる。

3．文法の定義法

本章では文法の定義法について次のように説明する。

まず文法を定義する時の最少単位となる構文要素について説明する。これはＳＥＭＡＣＳですでに定義済みのものであり、ユーザはこれを使って編集したい構文の文法を定義する。

次に文法の記述法について説明する。基本的にＢＮＦであり、そのＢＮＦ自体もＳＥＭＡＣＳでサポートしているので、ユーザはそれを使用して記述することができる。

最後に、作成した文法ファイルをＳＥＭＡＣＳが使うことができるかたちに変換する方法について述べる。

３．１　構文要素

ＳＥＭＡＣＳで定義済み構文要素を以下に示す。ユーザはこれらの構文要素を定義せずに文法の定義に使うことができる。ただし、これらの構文要素をユーザが定義し直すことはできない。

```
integer::=['-']Digit...
name::=Lowercase__letter[(Alphanumeric|'-')...]
Digit::=#0..#9
Lowercase __letter::=#a..#z
Alphanumeric::=Letter |Digit
Letter::=Lowercase__letter|Uppercase __letter
Uppercase __letter::=#A..#Z
```

さらに記法について説明しておく。

（１）＃がついているものは記号そのものを示す。

例　#aはa そのものを示す。

（２）「記号１．．記号２」は記号１から記号２までの（コードの）記号を示す。

例　#a..#zは英小文字を示す。

（３）「" 記号の列" 」は記号の列そのものを示す。その中に" を書きたい場合、" " と２つ書く。

例　'abc' は記号列abc 、''''は記号列 ' を示す。

（４）「記号. . . 」は記号を１個以上繰返し繋いだ記号列を示す。

例　Digit...は数字列を示す。

（５）「記号列１記号列２」は記号列１と記号列２とを繋いだ記号列を示す。

例　'ABC''DEF'は'ABCDEF'と同じである。

例　Letter Letter は英字２文字の列を示す。

（６）「記号１|記号２」は記号１または記号２であることを示す。

例　Letter|Digit は英字または数字を示す。

（７）「［記号列１］」は記号列１または空列を示す。

例　Letter [Alphanumeric...]は英字で始まる英数字列を示す。

（８）優先順（結合順）については、（４）～（７）の順に弱くなる。その順を変えるために () を用いる。

例

Digit... | Letter...	数字列と英字列（英字と数字の混在は不可）
(Digit | Letter)...	英数字列（英字と数字の混在可）
'a''b' | 'c'	文字列abか文字列c
'a'('b' | 'c')	文字列abか文字列ac

3．2　文法の記述法

（1）文法

基本的にはＢＮＦである。以下、本節では構造

if x=0 then y:=0 else y:=1・・・（ｉ）

とその文法（図３－１に示す）とをもとに説明する。なお（ｉ）を図３－１の文法と対応づけると図３－２のようになる。

```
top __category::=program.
program::=list(statement,';').
statement::=t(name), '=', expr |
     'if', test, 'then', statement, ['else', statement] |
     'while', test, 'do', statement |
     'read', t(name) |
     'write', t(name) |
     '(', program, ')'.
test::=expr, c__op, expr.
expr::=expr, op2, expr1 | expr1.
expr1::=expr1, op1, expr0 | expr0.
expr0::=t(name) | t(integer) | '(', expr, ')'.
c __op::='=' | '>' | '<'.
op1::='*' | '/'.
op2::='+' | '-'.
```

図３－１　文法の例

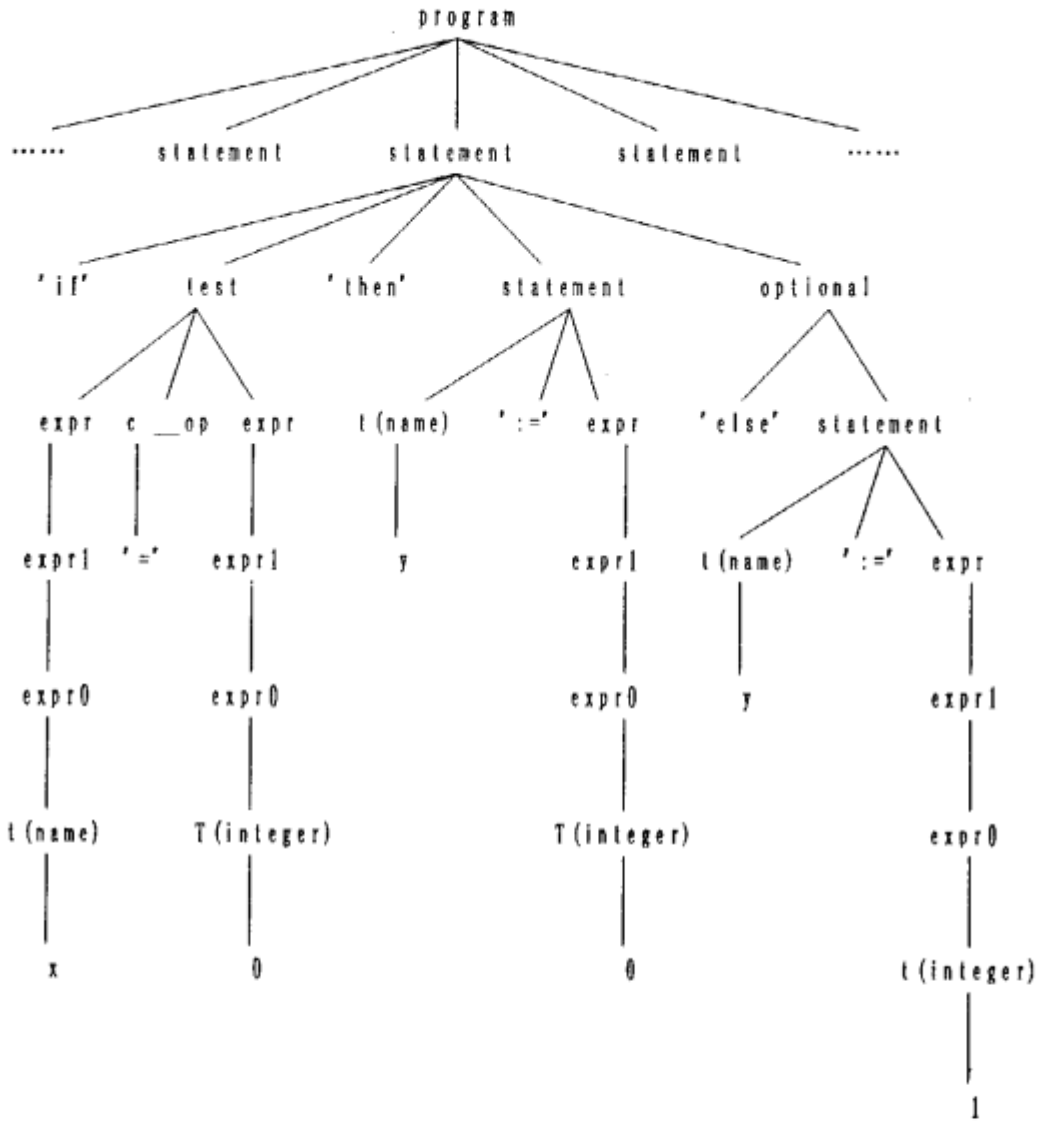


図３－２　（ｉ）の文法との対応づけ

①必ずtop __categoryでトップカテゴリを定義しておくこと。図３－１の例では

```
top __category::=program.
```

により、トップカテゴリがprogram であることを定義している。

②(a) list(X,Y) は、X がY で区切られて並ぶことを示す。図３－１の例では

```
program::=list(statement, ';')
```

により、program はstatement が；で区切られて並ぶものから成ることが示されている。

(b) 区切り記号がなしで並ぶ時は単にlist(X) と書く。

(c) listは右辺にしか現れない。また、その右辺において他の構文要素と並べて記述することはできない。たとえば、図３－３に示すような定義のしかたはできず、図３－４に示すように定義しなければならない。

```
function::=functor, ['(', list(function, ','),')'].
 functor::=t(name).
```

図３－３　間違った定義の例

```
function::=functor, ['(', argument__list,')'].
 functor::=t(name).
argument__list::=list(function,',').
```

図３－４　正しい定義の例

③t(X)は終端記号を、X は終端記号としてどんな構文要素がくるかを示す。

図３－１の例では、

```
t(name)
```

により、終端記号としてnameがくることを示している。

④''で囲まれたものはキーワードまたは記号を示す。図３－１の例では

```
'if', ':='
```

など。

⑤[]で囲まれたものはoptional（省略可能）であることを示す。図３－１の例では

```
['else', statement]
```

により、'else'と'statement' はoptionalであることを示す。

⑥そのほか、構文要素は、（カンマ）で区切り、１つ１つの定義の終わりには．（ピリオド）をつける。

（2）プリティプリント

プリティプリント用データを参照することにより、構文木データに対し改行、字下げなどを自動的に行って表示するのが、プリティプリント処理である。構造モードからストリングモードへ切り換えるときのほか、構造モード内で文法ガイダンス、ホロフラスティングなどによりエリアの表示が変化する場合にも使用される。

プリティプリント用データは、構文解析用データや文法ガイダンス用データのように自動生成されないため、新しい文法を定義するユーザが作成する必要がある。プリティプリント用データの定義は、あるカテゴリーとそのカテゴリーに対するサブカテゴリーやキーワードの関係で記述する。たとえば１５ページの文法に対して次のように表示させたいとする。

```
read B;
if A>B
  then C:=A
  else C:=B;
```

このためにはプリティプリント用データに、次のように記述する。

（ファイル名grm:expmple.ppi ）

```
  class pp-information-for-example has
    :pp-info(Class, Category, PP-info):-
        pp-info(Category, PP-info);
  local
  pp-info(top-category, [(program, list1)]);
  pp-info(statement, [(s(if), dum),
                      (test, dum),
                      (s(then), nltab(2)),
                      (statement, dum),
                      (s(else), nltab(2)),
                      (statement, dum)]);
  end.
```

list1, nltab(2), dumなどはプリティプリント用のキーワードであり、次のようなものが用意されている。

表３－１　プリティプリント用データのキーワードと処理内容

キーワード	処　　理　　内　　容
dum	直前に表示された構文要素の後に１文字分の空白を入れてから、その構文要素の表示を行う。記述のないカテゴリーに対してもこれと同じ操作を行う。
tab(N)	Ｎ文字分の空白を入れてから、その構文要素の表示を行う。
nltab(N)	改行し、親のカテゴリーの先頭の表示位置からＮ文字分の空白を入れてから、その構文要素の表示を行う。
list1	リスト要素の字下げを揃えて表示。区切り文字表示後に改行
list2	（テスト用）
list3	リスト要素の字下げを揃えて表示。改行後に区切り文字表示

3．3　文法定義ファイルの作成法

本節では構造編集時に必要なファイルの作成法について述べる。

（1）　debuggerのデフォルト・パッケージをsemacsとし、

```
:pre__parse(#parser __generator).
```

と入力する。

（2）　文法名を尋ねてくるので

```
Grammar. (CR)
```

と入力する。

（3）　論理名'grm:'の下に次の２つのファイルができていることを確認する。

```
'Grammar. idl'
'Grammar. gid'
```

4．構造の編集法

4．1　起動方法と画面の説明

（1）起動方法

システムメニューより起動する。

（システムメニューへの登録は、本書“2．インストール方法”を参照）

（2）ＳＥＭＡＣＳの画面

ＳＥＭＡＣＳが立ち上がった時は以下のウインドウを表示する。

（この時モードは＜ストリングモード＞である）

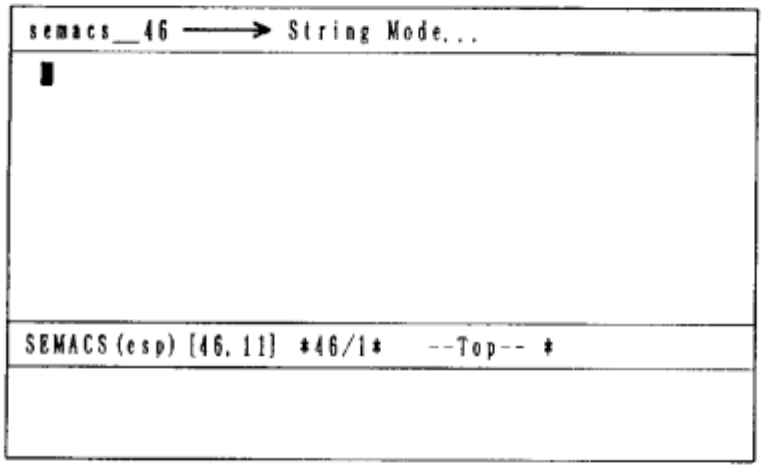


図４－１　ＳＥＭＡＣＳの初期画面

A．label__window

・プロセスＩＤ及びモードを表示する。

B．buffer__window

・編集を行う。

C．status__window

・ＰＭＡＣＳのstatus__windowと同じ。

D．echo__area__window

・ＰＭＡＣＳのecho__area__windowと同じだが、ＳＥＭＡＣＳのコマンドも受け付ける。

4．2　ストリングモード

（1）モード切り替え

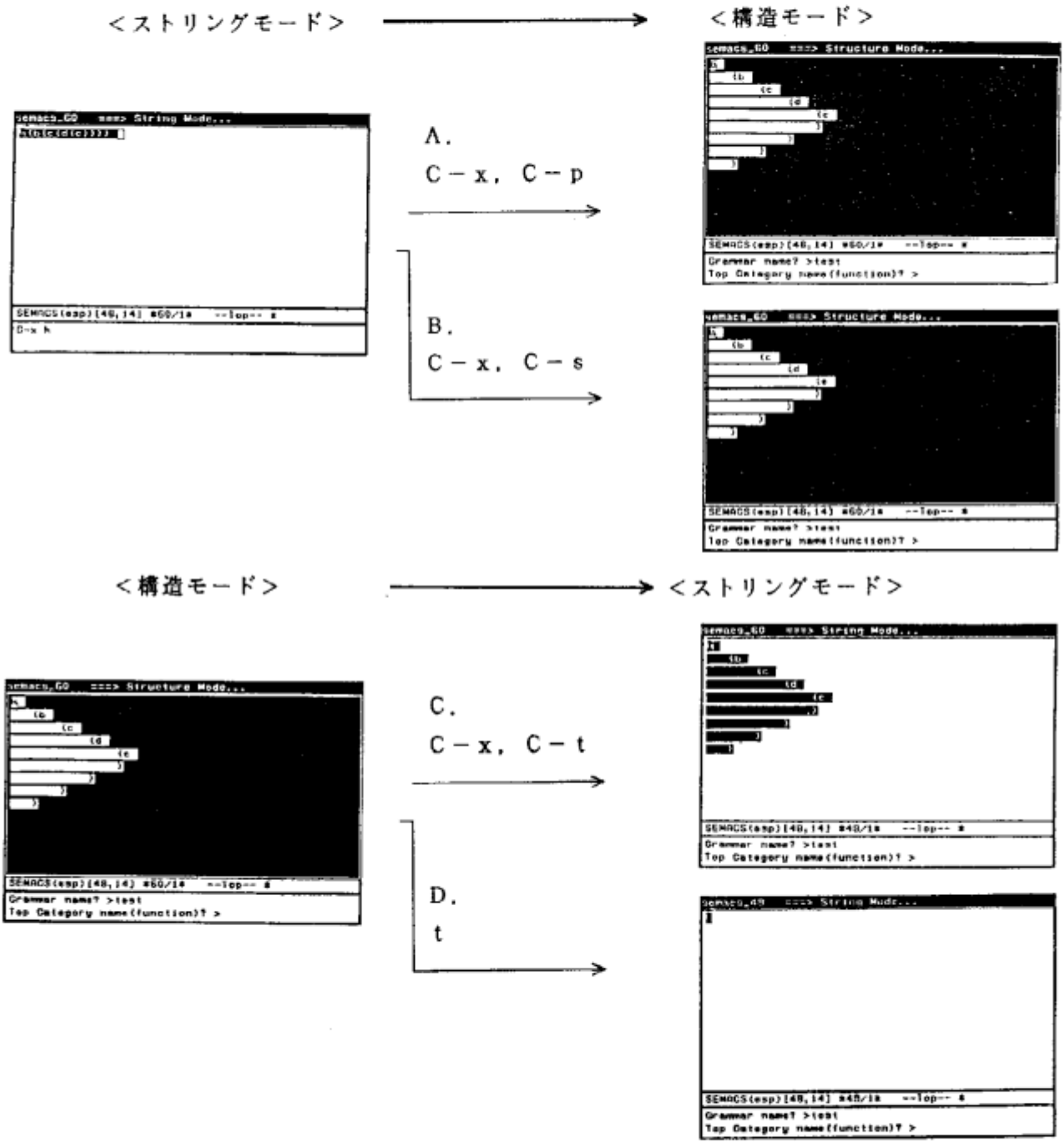


図4－2　モード切り替え

①. Ｃ－ｘ，Ｃ－ｐ（return__with__parse ）

ストリングモードから構造モードへの変換。

テキストをパーズして構造モードへ移行する。この時、まだ文法名とトップカテゴリ名が指定されていなければ、ユーザに問い合わせる。ただしリングモードでボックス以外を変更していても反映されない。

（次にストリングモードに返ってきたときに変更前のイメージにもどっている）

②. Ｃ－ｘ，Ｃ－ｓ（return__without __parse ）

ストリングモードから構造モードへの変換。

テキストのパーズはしない。この時、構造モードではストリングモードに移るまえの編集画面が表示される。従って最初にストリングモードで編集対象を作成して構造モードへ移るときは、このコマンドを使ってはならない。

（もしこのコマンドを使った時は、それまで作成した編集対象がクリアされてしまう）

③. Ｃ－ｘ，Ｃ－ｔ（goto__simple__string__mode）

構造モードからストリングモードへの変換。

この時、構造モードで変更された箇所はストリングモードに反映される。

④. ｔ（goto__simple__string__with__delete）

構造モードからストリングモードへの変換。

この時、エリアを空にしてストリングモードへ移る。

（２）編集機能

通常のテキスト編集機能を持つ。編集機能はＳＩＭＰＯＳのＰＭＡＣＳと同様である。ただし、構造モード時のボックスに対応する領域が反転表示される。ボックス内を変更した場合、return__with__parse によって構造に反映されるが、ボックス外を変更しても反映されない。

図４－３　ストリングモード時のＳＥＭＡＣＳ

（3）ボックス領域の再セット

ボックス領域を変更したい場合、以下の操作でボックス領域を変更できる。本コマンドの使用に関してはＳＥＭＡＣＳに対する充分な理解が必要である。

（ａ）．変更したい領域の先頭をマークする。

（ｂ）．変更したい領域の末尾にカーソルを移動する。

（ｃ）．Ｃ－Ｍ－ｂ押下。

図４－４　ボックス領域の再セットの方法

（注意）構造上での現在位置およびトップカテゴリーは変らない。

4．3　エリアの移動

（1）概要と使用例

編集対象（木構造をしている）のうち今編集している部分（部分木になっている）がエリアである。エリアの移動は、テキスト・エディタにおけるカーソルの移動に対応する。エリアは部分木から部分木へ移動するため、以下のコマンド群は木構造の中で部分木を渡り歩くためのものになっている。いずれのコマンドにおいても、移動が不可能なときはエリアは元に状態を保つ。

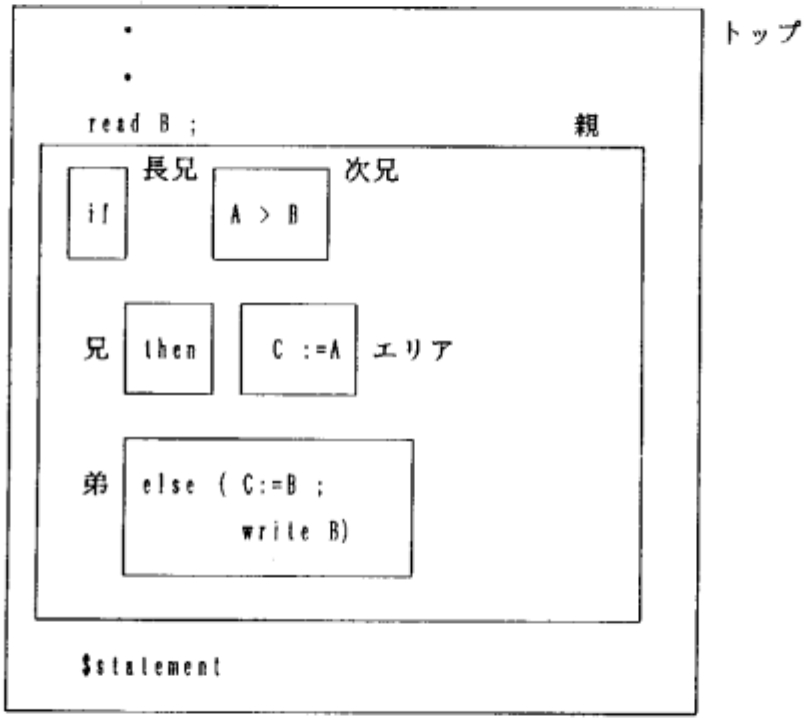


図４－５　エリアの移動

使用例

上図のようなテキストがあり、エリアが C :=Aの上にあるとき、各コマンドによりエリアは以下のように移動する。（図３－１に示した文法を使用した場合）

コマンド	エリア	図との対応
(a) top	テキスト全体	トップ
(b) parent	if A > B then C :=A else (C :=B; write B)	親
(c) child	C	

```
(d) youngest-child                A
(e) elder-brother                 then                          兄
(f) younger-brother       else ( C :=B ; write B)               弟
(g) eldest-brother                if                            長兄
(h) yongest-brother       else ( C :=B ; write B)               弟
(i) 1                             C
    2                             :=
    3                             A
```

（2）コマンド

①. C-a （top ）

テキスト全体をエリアにする。

②. C-p;key#up;mouse##ll （parent）

現在のエリアを含む親の部分木を新しいエリアにする。ただし、兄弟のない部分木はエリアにならない。

③. C-n;key#down （child ）

現在のエリアに含まれる子供の部分木のうちで、最も左にある部分木を新しいエリアする。ただし、兄弟のない部分木ははエリアにならない。

④. M-n （youngest-child）

現在のエリアに含まれる子供の部分木のうちで、最も右にある部分木を新しいエリアする。ただし、兄弟のない部分木ははエリアにならない。

⑤. C-b;key#left （elder-brother ）

現在のエリアと兄弟関係にある部分木のうちで、エリアの左となりにある部分木を新しいエリアにする。

⑥. C-f;key#right （younger-brother ）

現在のエリアと兄弟関係にある部分木のうちで、エリアの右となりにある部分木を新しいエリアにする。

⑦. M-b （eldest-brother）

現在のエリアと兄弟関係にある部分木のうちで、最も左にある部分木を新しいエリアにする。

parent＋child と同じ。

⑧. M-f （youngest-brother）

現在のエリアと兄弟関係にある部分木のうちで、最も右にある部分木を新しいエリアにする。

parent＋youngest-childと同じ。

⑨. 1 ;...;9 (n-child)
入力された数字をＮとするとき、現在のエリアに含まれる子供の部分木うち左から数えてＮ番目の部分木を新しいエリアにする。

⑩. mouse##1 (move-area-with-mouse-pointing)
マウスで指定された点を含む最少の部分木を新しいエリアとする。

4．4　エリアの削除、復元

（1）概要と使用例

部分木の削除と、削除された部分木の復元をおこなう機能である。これにより、部分木の移動、複写が可能になる。

削除した部分木とそれを復元できるエリアには構造上の制約がある。例えば、

```
expr
|
expr1
|
expr0
|
t(A)
```

という部分木は、area-top-category がexpr、expr1 またはexpr0 のときはyankできるが、それ以外の場合はyankできない。（yankは削除された要素を復元するコマンド）

使用例

```
    :
if A > B then C:=A else ( C :=B ; write D )
```

のとき、上記の部分木をyankしようとすると

エリアがA またはB ならばyankできる。

エリアが、if、A > B 、(C :=B ; write B) などのときにはyankできない。

（2）コマンド

①. C-w （kill-area ）

エリアの内容をarea-top-category だけ残して削除し、削除した内容をyank用のスタックに保存する。エリアのあった位置には 'S' area-top-category と表示される。ただし、area-top-category が'optional'の場合はarea-top-category も残さないし、yank用のスタックにもその内容は保存されない。

②. C-y （yank-area ）

yank用のスタックからエリアを一つ取り出し、現在のarea-top-category に接続可能かどうかを調べる。可能であれば現在のエリアと取り替え、可能でなければなにもしない。現在のエリアがリストの要素の場合には、yankされたエリアは（現在のエリアと取り替えられるのではなく）兄の位置に挿入され、そこが新しいエリアになる。

③. C-d （delete-area ）
現在のエリアを削除するが、削除されたエリアはyank用スタックに保存されない。

④. M-w （put-area）
現在のエリアをyank用のスタックに保存するが削除はしない。

⑤. M-y （meta-yank-area）
yank用のスタックをさかのぼってyank-area する。meta-yank-areaは有効な場合にしか作用しない。meta-yank-areaが有効になるのは次の二つの場合だけである。

yank-area の直後に入力した場合、または、
有効なmeta-yank-areaの直後に入力した場合

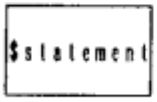
4．5　構造の探索、置換

（1）概要と使用例

構造に関する情報を使用する探索及び置換であり、単に文字列だけに注目したものに比較した場合、より目的に適した操作が可能である。

使用例（reverse-replace-area）

```
    :
    :
read B ;
if A > B then C := else (C := B ; write B );
```



（エリア）

```
    :
    :
```

という状態で、コマンドreverse-replace-areaを続けて呼ぶと、以下のようにエリアの内容が変化する。

(a) write B
(b) C := B
(c) (C := B ; write B)
(d) C := A
(e) if A > B then C :=A else (C :=B ; write B)
(f) read B

（2）コマンド

①. s （tree-search ）

'category>' というプロンプト後に探索する非終端記号名を入力する。その時点のエリアの後方に、その非終端記号名を根とする部分木を探索し、見つかったときはその部分木にエリアを移動させる。

②. r （reverse-tree-search ）

'category>' というプロンプト後に探索する非終端記号名を入力する。その時点のエリアから逆方向に、その非終端記号名を根とする部分木を探索し、見つかったときはその部分木にエリアを移動させる。

（この２つの探索が連続して実行されるときは、前回の非終端記号名が探索の対象となる。またプロンプトの直後に、retrun-keyが入力されると、探索に使用された最新の非終端記号名が探索される。）

③. C-s （pattern-search）
'I-search:' というプロンプト後に探索する文字列を入力する。その時点のエリアの後方に入力した文字列を探索する。探索はＰＭＡＣＳのインクリメンタル・サーチと同様に行う。探索終了キー（ＥＳＣキー）入力すると、見つかった文字列を含む部分木にエリアを移動する.

④. C-r （reverse-pattern-search）
'Reverse i-search:' というプロントプト後に探索する文字列を入力する。その時点のエリアから逆方向に、入力した文字列を探索する。探索はＰＭＡＣＳのリバース・インクリメンタル・サーチと同様に行う。探索終了キー（ＥＳＣキー）を入力すると、見つかった文字列を含む部分木にエリアを移動する。

⑤. M-s （new-tree-search ）
探索コマンド実行直後、異なる非終端記号名探索するときに使用する。

⑥. M-r （return-before-searching ）
一連の探索操作以前の位置にエリアを戻す。

⑦. M-/ （replace-area）
現在のエリアの根と同じ非終端記号名を根とする部分木を、構文木の先頭から探索し、その部分木でエリアを置き換える。

⑧. / （reverse-replace-area）
現在のエリアの根と同じ非終端記号名を根とする部分木を、エリアの前方を逆向きに探索し、その部分木でエリアを置き換える。

⑨. C-x,r （return-before-replace ）
エリアをreplace 以前の状態に戻す。

4．6　リスト要素の処理

（1）概要と使用例

木構造における兄弟がリストの構造をしているとき、リストの要素を操作するための機能である。

使用例

```
    :
  if A > B then C :=A else ( C :=B ; write B )
```

のとき、エリア C := B ; write B において各コマンドは以下のように作用する。

insert-forward　→　C :=B ; write B ; $statement（下線部がエリア）

insert-backward →　$statement ; C :=B ; write B（下線部がエリア）

（2）コマンド

①. C-x, f　(insert-forward)

リスト全体がエリアの場合はその子の末弟の位置に、リスト要素がエリアの場合はその弟の位置にそれぞれのリスト要素のcategoryのみから成る部分木を挿入し、それを新しいエリアとする。

②. C-x, b　(insert-backward)

リスト全体がエリアの場合はその長男の位置に、リスト要素がエリアの場合はその兄の位置に、それぞれそのリスト要素のcategoryのみから成る部分木を挿入し、それを新しいエリアとする。

③. C-x, C-d　(delete-area-element)

リスト要素がエリアのとき、そのエリアを削除する。area-top-category まで削除される点はoptionalのkillと同じだがその内容はrank用スタックに保存される。削除の結果、リストの要素が一つもなくなる場合は例外で、kill-areaと同様の結果となる。

4．7　ホロフラスティング

（1）概要と使用例

構造を巨視的に見たり、微視的に見たりするための機能である。コマンドholophrastが呼ばれると、エリアの表示はそのトップカテゴリーのみとなる（$$area-top-category と表示される）。しかしエリアの削除と異なり、エリアの構造の情報は保存されているので、unholophrastあるいはone-level-unholoを使用することによりエリアの表示は復元される。

使用例

：

：

```
if A > B then C :=A else ( C :=B ; write B )
```
（エリア）

この状態から各コマンドを実行したときのエリアの表示の変化を以下に示す。

	コマンド	エリア
(a)	holophrast	$$statement
(b)	one-level-unholo	if $$test then $$statement $$optional
(c)	unholophrast	if A > B then C :=A else (C :=B ;write B)

（2）コマンド

①. C-h （hlophrast ）

エリア全体をホロフラスティング

②. C-e （one-level-unholo）

ホロフラストのレベルを１段低くする。

③. C-x, C-h （holophrast-with-level ）

入力されたレベルでエリア内をホロフラスティング

④. C-x, C-u （unholophrast）

エリア内のホロフラスティングを解除

4．8　文法案内

（1）概要と使用例

ある非終端記号の展開の可能性を示し、そのいずれかをメニューから選択させて構文木を展開する。文法に不慣れなユーザ向けの機能であるが、キーワードの打ち間違いが防げるなど文法に習熟したユーザにとっても利用価値のある機能である。

使用例

トップカテゴリーがstatement のときコマンドguide を実行すると次のような、文法ガイダンスウインドウが表示される。このウインドウはメニューウインドウになっており、マウスクリックの種類により選択したサブカテゴリーでトップカテゴリーを展開した木を新しいエリアとする、あるいはそのサブカテゴリーのさらに詳細な情報をみる、の２種類の操作を行うことができる。

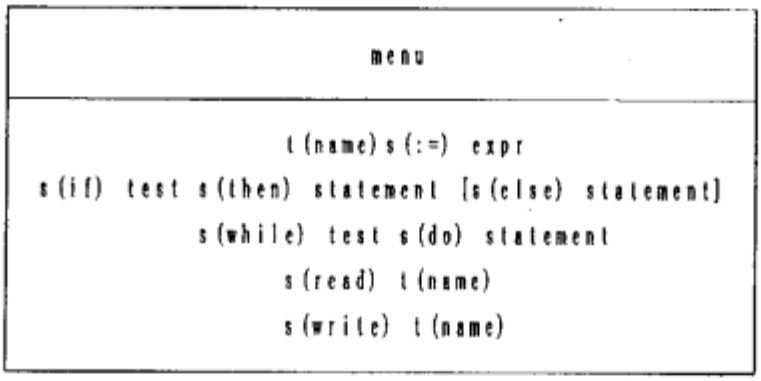


図４－６　構文ガイダンスウインドウ
（s( )はキーワードまたは記号を表す）

（2）コマンド

①. C-x, g （guide ）

エリアのtop-categoryを展開したメニューを表示する。

そのメニュー上で項目をmouse中ボタンでクリックするとその展開結果にエリアを置き換える（もとのエリアの内容は削除される）。それ以外のクリックで項目を選択すると、only-detail-see-modeとなる。

②. C-x, n （nt-guide）

エコー・ウインドウから入力されたcategoryを展開したメニューを表示する。この時点で既にonly-datail-see-modeであり、エリアの内容には影響を与えない。

only-detail-see-mode選択された項目がa, b, c から成るとすると、まずa を

展開したメニューを表示する。（a がterminal-category, a( ), T( )の場合は、b に制御が移る）。そのメニューで、ある項目が選択された場合はその項目に関して（再帰的に）、ここで記述された操作をしたのち、b, cに関して同じ操作を行う。マウスがウインドウから外された場合は、単にb, cに関する操作を行う。

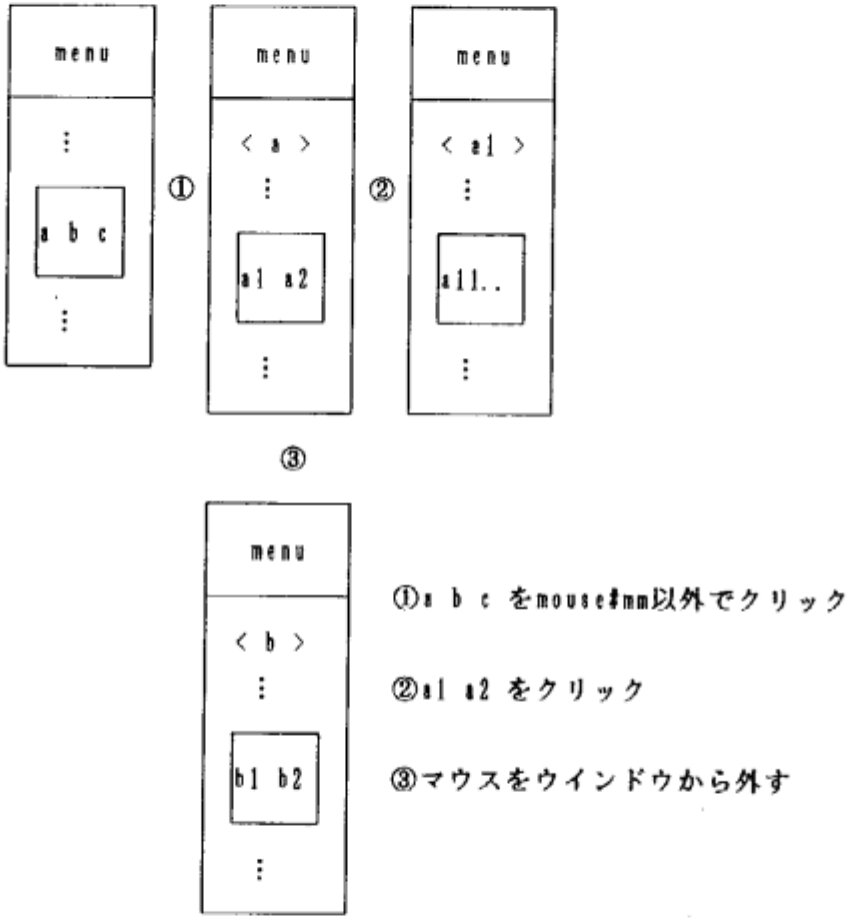


図４－７　only-detail-see-mode時のウインドウ変化

4．9　バッファの切り替え

（1）概要

ＰＭＡＣＳと同様にバッファを切り替えることで、編集対象を変更できる。ただしＳＥＭＡＣＳではストリングモードと構造モードが明確に分かれているためバッファを切り替えた場合、それぞれが実行されていたモードに自動的に切り替える。

（2）操作

ＰＭＡＣＳと同様である。（ただし構造モード時は、マウスクリックｍｍは使用出来ない）

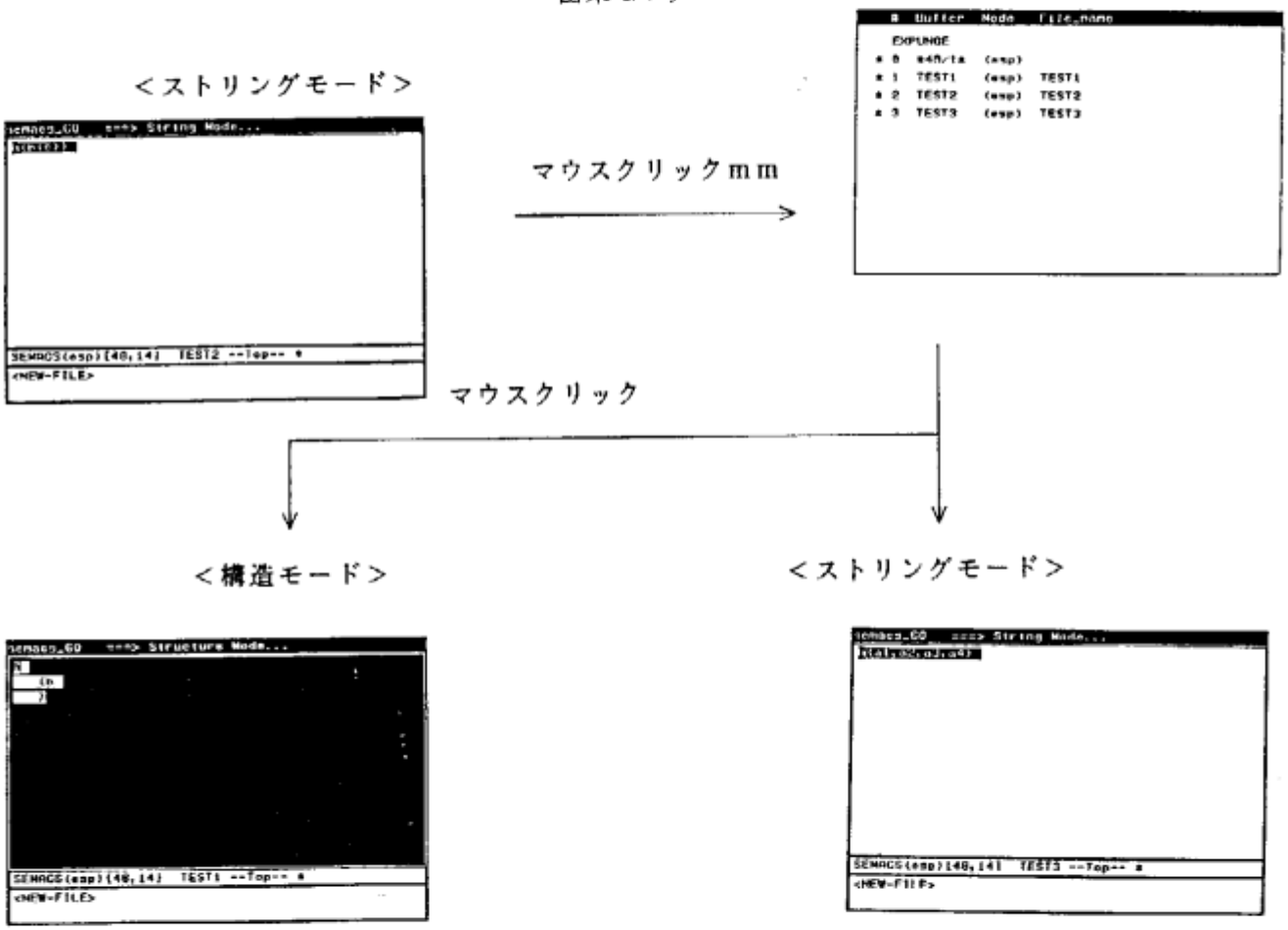


図４－８　ストリングモード時のバッファの切り替え

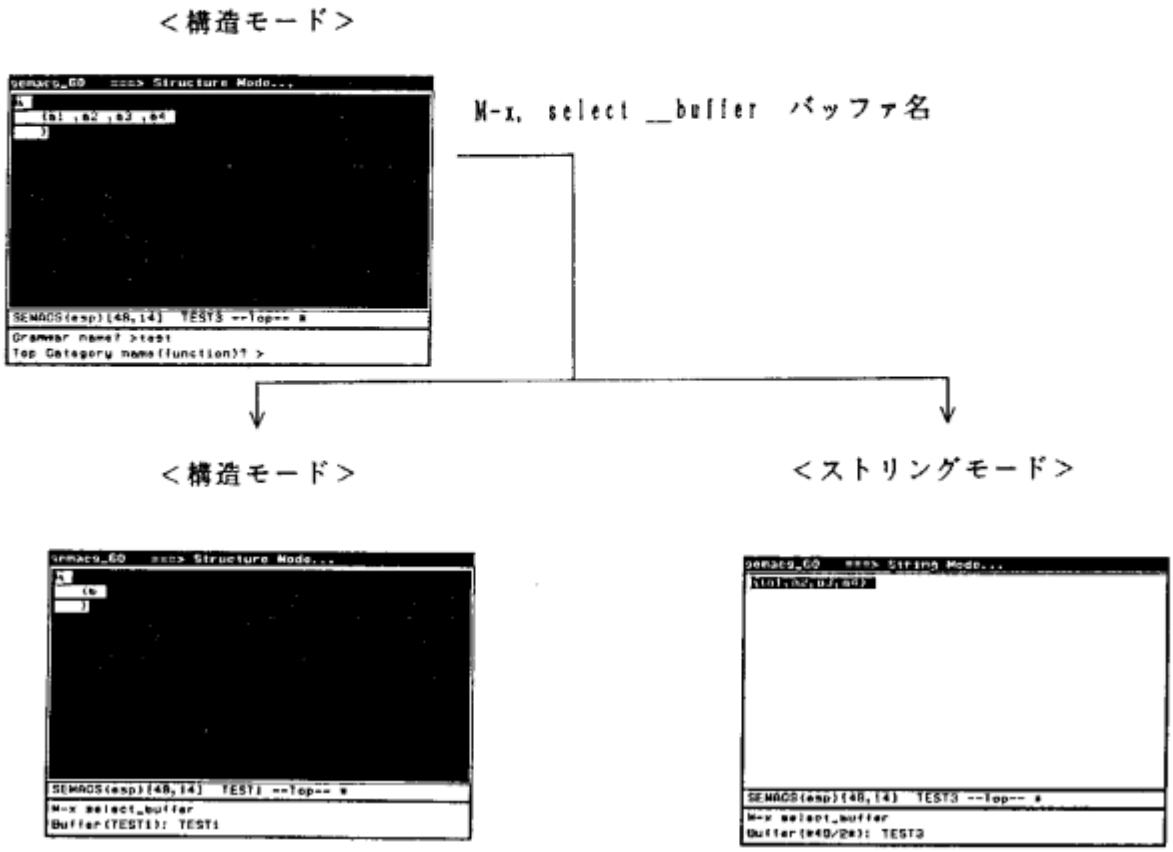


図４－９　構造モード時のバッファの切り替え

4．10　セーブ、ロード

ストリングモード及び構造モード時に編集対象をファイルのセーブ、ファイルからロードできる。

操作は基本的にＰＭＡＣＳと同様であるが、save_with_dateは使用できない。

（1）セーブ

その時のモードに応じてファイル形式が異なる。

ストリングモード	ストリングイメージのファイル
構　造　モ　ー　ド	構造データのファイル

（2）ロード

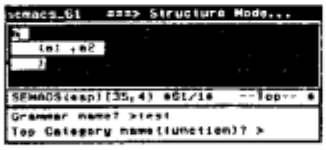
ロードするファイルの内容によりモードを変更する。

ストリングイメージのファイル	ストリングモードへ移行
構造データのファイル	構造モードへ移行

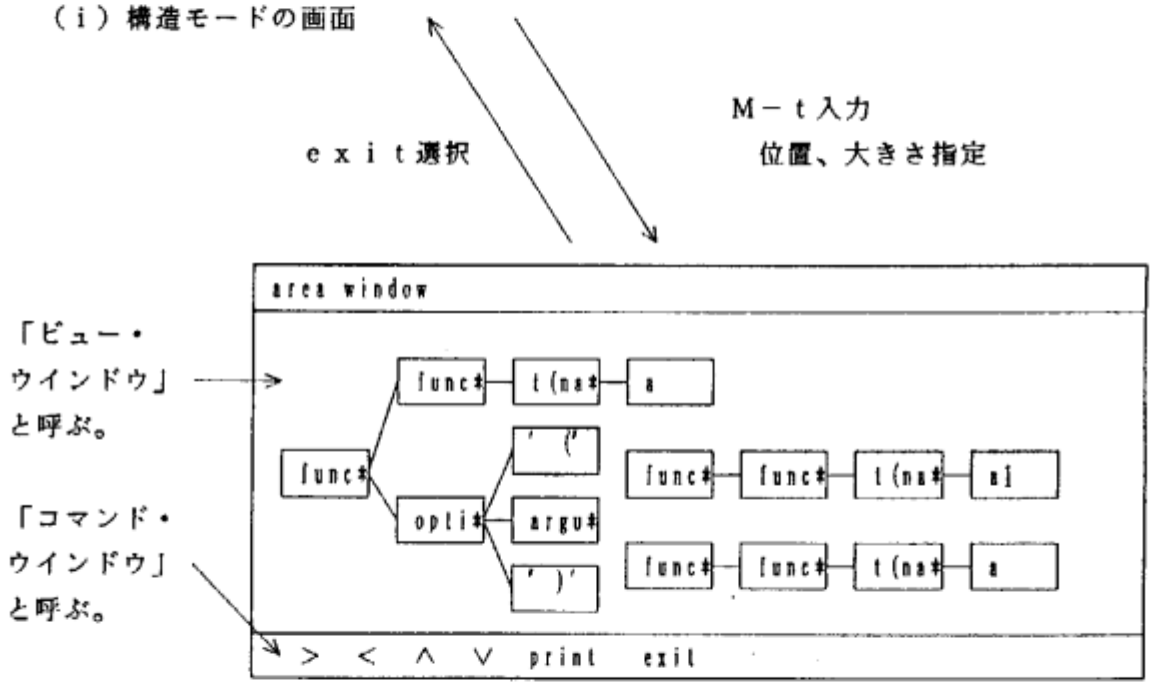
5．木表示機能

5．1　エリアの木表示

木表示機能の基本的機能はその時のエリアを木表示する機能である。図５－１に例を示す。



（ｉ）構造モードの画面



（ii）木表示ウインドウ

図５－１　エリアの木表示

位置、大きさを指定するのは初めてウインドウを生成する時だけである。

構造モードの画面への戻り方は２通りある。

図５－１の木表示ウインドウのコマンド・ウインドウでexitを選択すると、図５－２に示すexitウインドウが表示される。このexitウインドウで「はい」を選択す

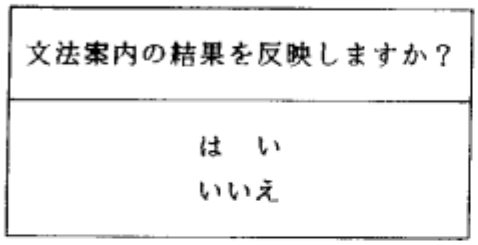


図５－２　exitウインドウ

ると文法案内（５．５で説明する）の結果が構造モードの画面に反映される。「いいえ」を選択すると木表示を行う前の構造モードの画面に戻る。

木表示中には構造モードの画面には入力することはできない。

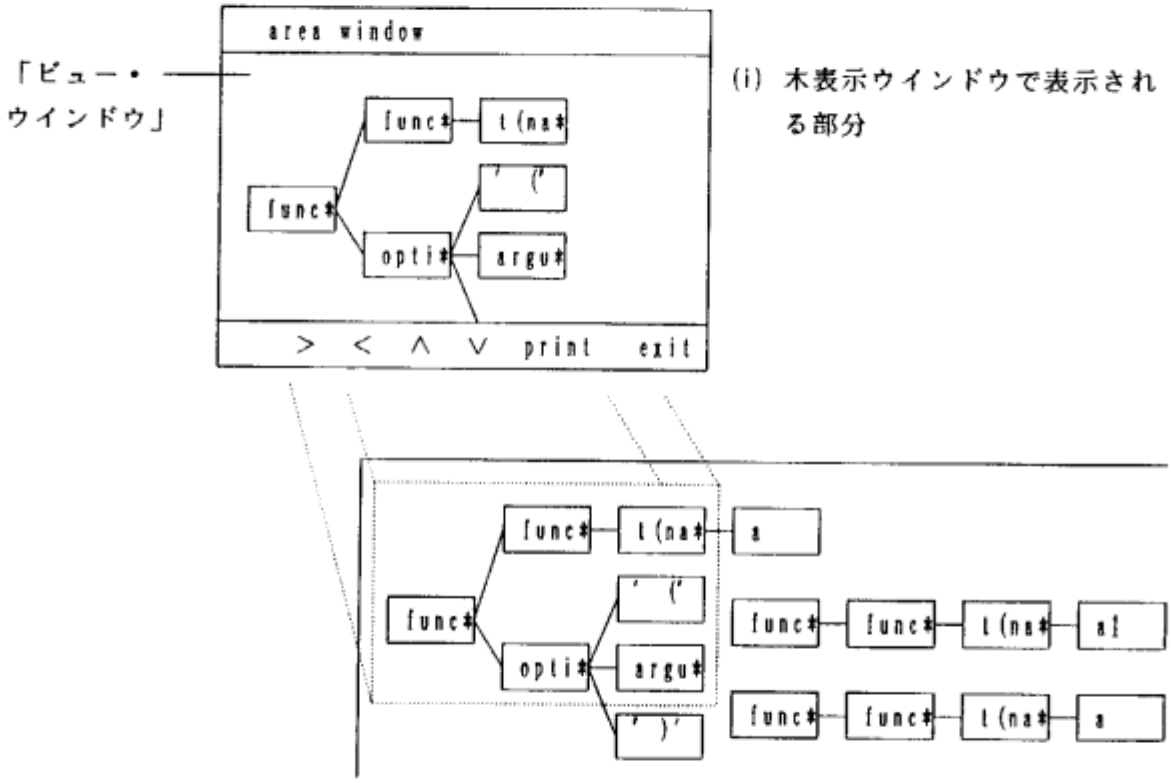


図５－３　ビュー・ウインドウとワールド・ウインドウ

木表示ウインドウには、ユーザから見えないウインドウとしてワールド・ウインドウがある。木はまずワールド・ウインドウに描かれる。そして、ワールド・ウインドウの一部をビュー・ウインドウにコピーしてユーザに表示する。

ワールド・ウインドウに描くことができるノードの最大個数は

深さの方向　１４個（１５個目が途中で切れる。）

幅の方向　　６０個（６１個目が途中で切れる。）

である。

コマンド・ウインドウでprint を選択した場合はワールド・ウインドウの左上隅からスクリーンと同じ大きさの部分のハード・コピーがとれる。

５．２　ノードの名前表示

ノードの名前が６文字以上の場合、図５－４に示すように５文字目が＊となってそれから先が省略されていることが示される。

ノード上でmouse#1 を入力すると、名前が７７文字以下の場合はフルネームがテンポラリ・ウインドウで表示される。７８文字以上の場合は、７７文字目が＊となって省略されていることが示される（７８文字以上の場合にフルネームを表示することはできない）。

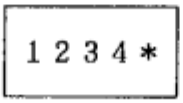


図５－４
ノードの名前表示

5．3　ビュー・ウインドウのスクロール

ビュー・ウインドウはマウスを使って上下にスクロールすることができる。その他にコマンド・ウインドウの項目をマウスでシングル・クリック（左、真ん中、右いずれも可）することにより、スクロールすることができる。

＞で右にかくれている部分が、
＜で左にかくれている部分が、
∧で上にかくれている部分が、
∨で下にかくれている部分が、

それぞれ表示される。１回のクリックでスクロールされる大きさは、右、左、上、下それぞれの方向の一番遠く離れたノードがはいる位置までである（図５－５参照）。

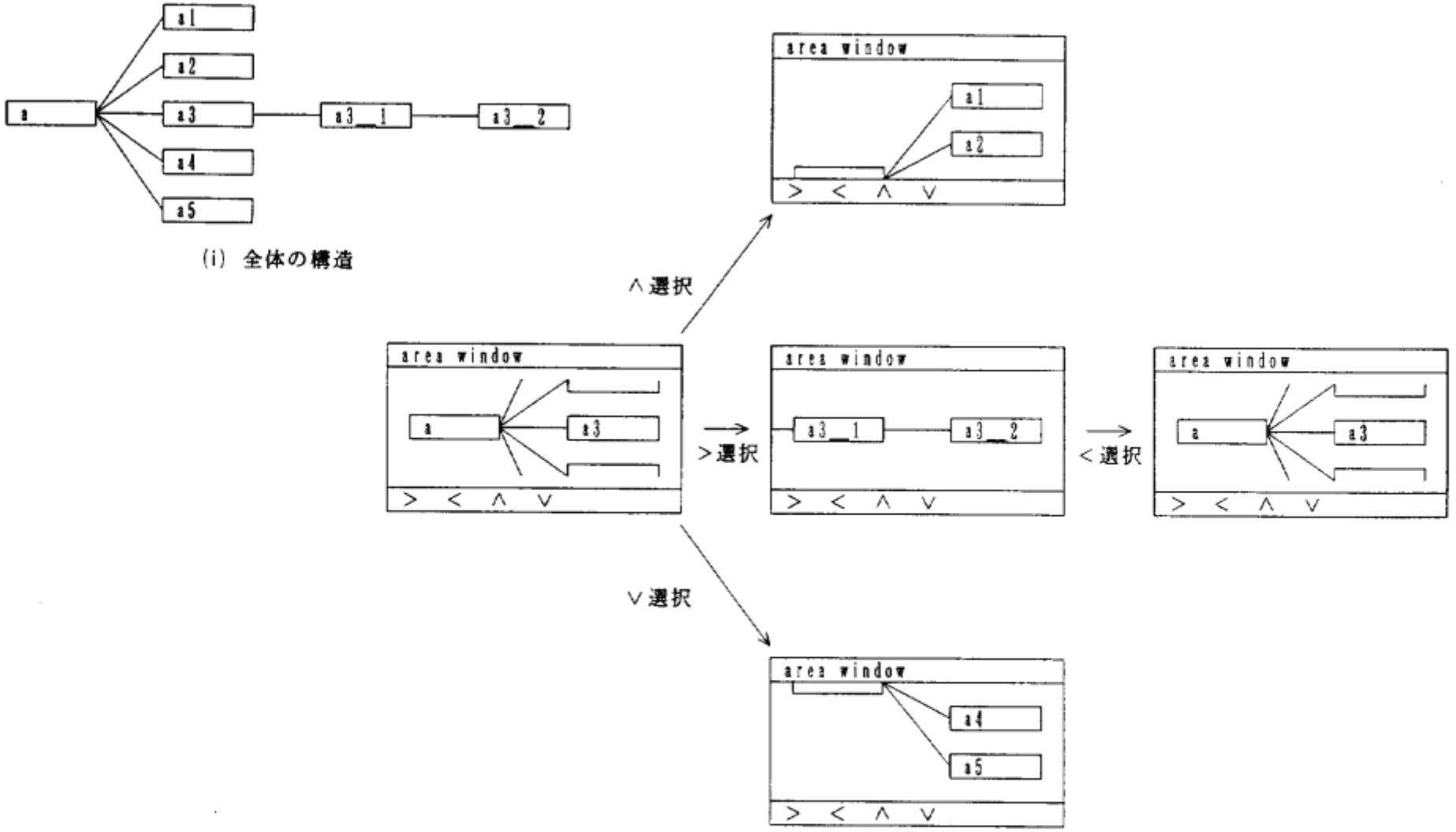
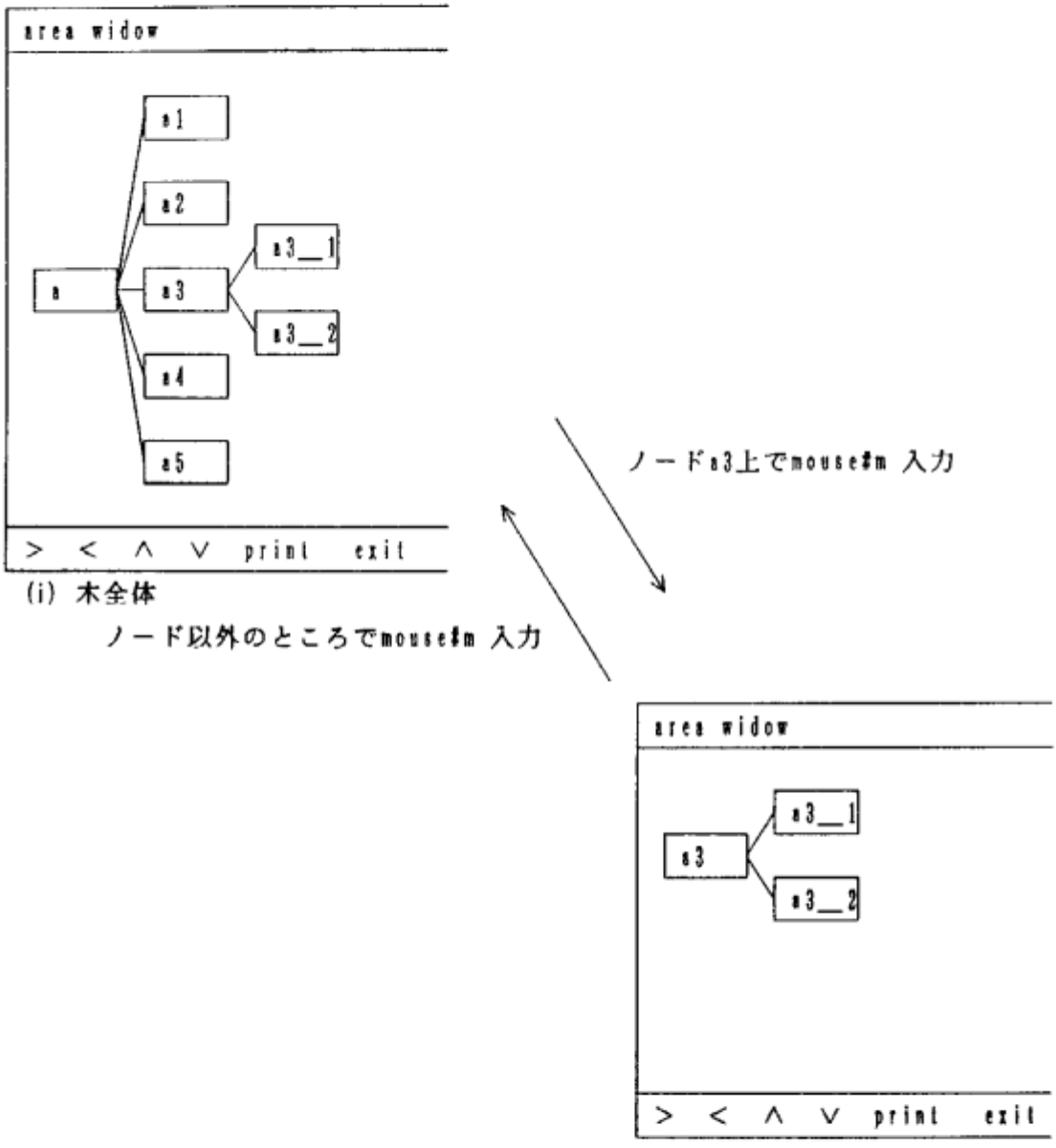


図５－５　コマンド・ウインドウの項目選択によるスクロール

5．4　ユーザが選択したノードをルート・ノードとする部分木の表示

図５－６に例を示す。



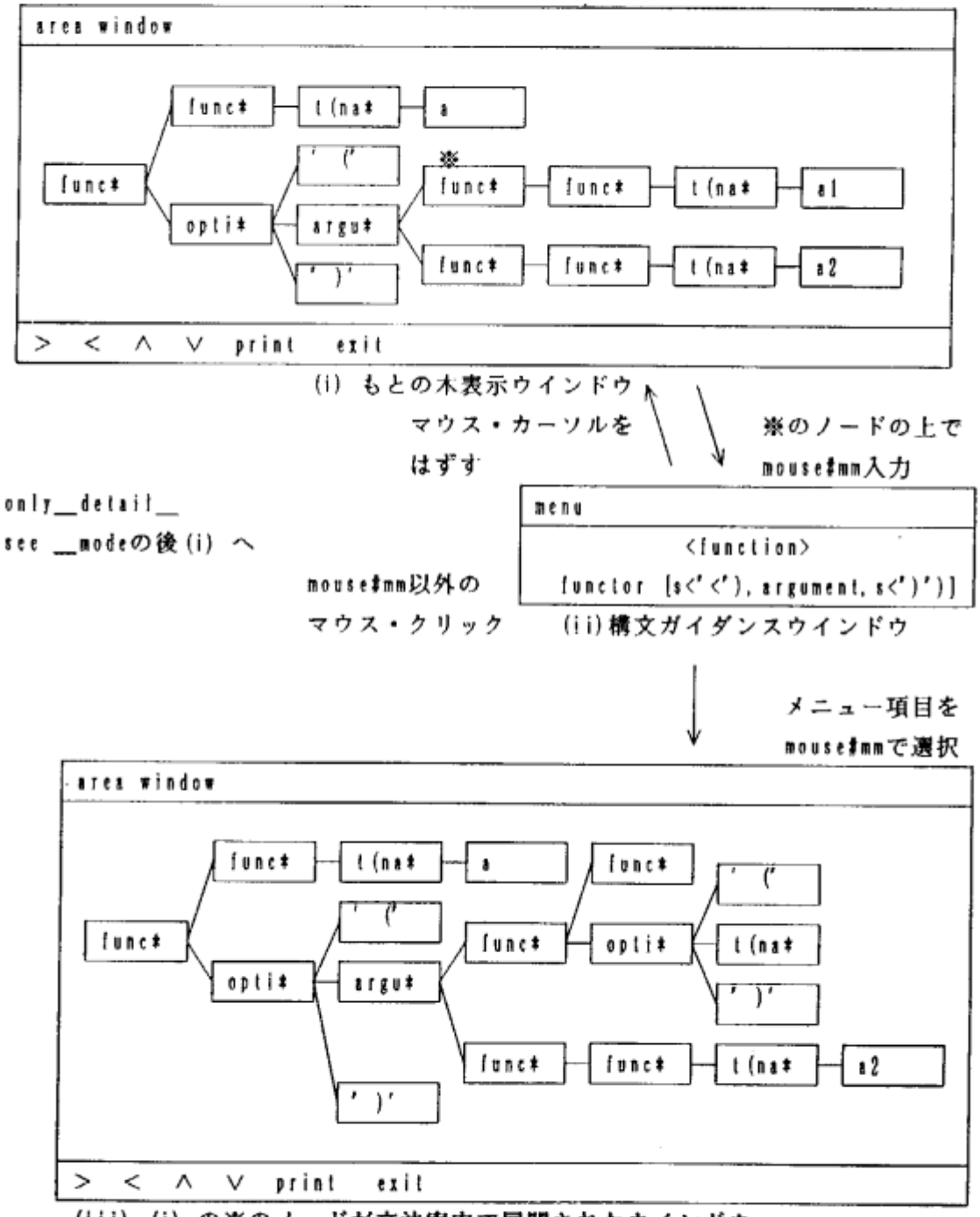
(i) 木全体

(ii) a3をルート・ノードした部分木の表示

図５－６　ユーザが選択したノードをルート・ノードとする部分木の表示

5．5　文法案内

図５－７に例を示す。



(iii) (i) の※のノードが文法案内で展開されたウインドウ

図５－７　文法案内の例

```
function :: = functor, ['(', argument,')'].
functor  :: = t(name)
argument :: = list(function,',').
```

図５－８　図５－７で使われている文法

表５－７　文法案内不可のノード

キーワード（含　記号）、optional、終端記号、t(name) t(integer)

図５－９　optionalの場合の構文ガイダンス・ウインドウ

表５－１に示すノード上では文法案内を行わない。ただし、optionalの場合のみ図５－９のような構文ガイダンス・ウインドウが表示されるが、展開することはできない。また、only_detail_see _modeもない。

付録　構造エディタコマンド一覧（構造モード時）

基本的に構造モード時はＰＭＡＣＳコマンドは使えず、構造編集コマンドのみが使用できる。

ただし幾つかのコマンドはＰＭＡＣＳと同様に使用できる。

M-x, コマンド名 ……	ただしここで使えるコマンドは以下のものだけである。
キーアサイン無し	(other__window)
C-v	(view__next__screen)
M-v	(view__previous__screen)
r	(return__box) ……新規作成ボックスの先頭から表示する
C-x, C-w, ファイル名	(write__file)
C-x, C-v, ファイル名	(visit__file)
C-x, C-f, ファイル名	(find__file)
キーアサイン無し	(select__buffer)

次頁に構造エディタコマンド一覧（構造モード時）を示す。

構造エディタコマンド一覧（構造モード時）

コ　マ　ン　ド	キ ー ・ ア サ イ ン
エリアの移動	
parent	C-p ; key#up ; mouse##ll
child	C-n ; key#down
youngest-child	M-n
n-child	1;2;...;9 ; C-z
top	C-a
elder-brother	C-b ; key#left
younger-brother	C-f ; key#right
eldest-brother	M-b
youngest-brother	M-f
move-area-with-mouse-pointing	mouse#l
エリアの削除・復元	
delete	C-d
kill	C-w
put	M-w
yank	C-y
meta-yank	M-y
リスト要素の挿入・削除	
insert-forward	C-x, f
insert-backward	C-x, b
delete-area-elements	C-x, C-d
文法ガイダンス	
guide	C-x, g ; mouse#mm
(sub-commands)	
expand	mouse#mm(on menu window)
detail-display	other mouse clisks
nt-guide	C-x, n

構造エディタコマンド一覧（構造モード時）

コ　マ　ン　ド	キ　ー　・　ア　サ　イ　ン
ホロフラスティング	
holo	C-h
holo-with-level	C-x, C-h
unholo	C-x, C-u
one-level-unholo	C-e
探索・置換	
pattern-search	C-s
reverse-pattern-search	C-r
tree-search	s
reverse-tree-search	r
return-before-searching	M-r
new-tree-search	M-s
replace-area	M-/
reverse-replace-area	/
return-before-replace	C-x, r
モード切り替え	
return-with-parse	C-x, C-p (in string-mode)
return-without-parse	C-x, C-s (in string-mode)
goto-string-mode	C-x, C-t
goto-string-mode-with-delete	t
画面スクロール	
next-window	C-v
previous-window	M-v
return-box	v