逐次型推論マシンＰＳＩ上の汎用構造エディタＳＥＭＡＣＳ

Grammar-free Structure Editor SEMACS on PSI

小久保岩生[1]，藤田正幸[1]，高谷喜一[1]
梶山拓哉[1]，吉武厚[2]，坂井公[3]，横田一正[4]
[1]㈱三菱総合研究所　[2]㈱アーティフィッシャル・インテリジェンス　[3]三菱電機㈱
[4]新世代コンピュータ技術開発機構

逐次型推論マシンＰＳＩ上のテキストエディタ pmacsの機能を利用した汎用構造エディタSEMACS(Syntax-directed Extension of pMACS)について報告する．SEMACSには，文字列に対して編集を行うモードと構文木に対して編集を行う構造モードの２つがあり，ユーザはその両モードを適宜使い分けて使用することができ，文法に対する汎用性がある．本報告では，SEMACSの特徴，機能，および応用例について説明する．

1．開発の目的

ソフトウエア・システムのうち，ユーザによるテキストの入力・修正作業が多いものは，その作業を支援する高度なユーザインタフェースが必要となる．文字列を編集の対象とするテキストエディタに対して，言語の構造を編集の対象とするエディタは構造エディタ[1][2]と呼ばれ，そのようなユーザインタフェースを実現する有効な手段となる．テキストエディタと比較した場合の構造エディタの長所として次のことが考えられる．

①編集コマンドの高度化・・・エディタが対象言語の構造に関する知識を持っているため，探索，置換，削除，復元といったコマンドの内容をテキストエディタより高度にすることができ，編集中のテキストの文法的な妥当性を維持することができる．また文法に不慣れなユーザに文法に関するガイダンスを行ったりする処理も可能となる．

②対象システムとの有機的結合・・・一般にテキストエディタで作成したテキストを何らかの意味で実行する際の第一段階となる構文解析が不必要となり，修正後，再び構文解析する場合も修正箇所のみを処理すればすむ可能性がある．また実行箇所のリアルタイム表示，実行中の部分的な修正といった操作が可能となる．

そのような長所があるにもかかわらず，テキストエディタに比べ構造エディタで広く使用されているものが少ない理由として，次のことが考えられる．

①言語，システムへの依存性・・・対象とする言語またはシステムに対する依存性が高いと，他の言語やシステムで直接利用したりその技術を移植することが難しく，また対象言語自身の変更にも弱くなる．

②操作性の悪さ・・・文法に無関係に自由な記述が行えるテキストエディタに比べ操作上の制約が多く，行いたい操作が円滑に行えない．

以上を考慮し，複数のシステムで利用できる使いやすいインタフェースの核となることを目的として，㈱新世代コンピュータ技術開発機構(ICOT)の Kappaプロジェクト[3][4]と CAPプロジェクト[5]の共同で，SEMACS(Syntax-directed Extension of pMACS)[6]は開発された．

2．特徴

SEMACSは，ICOTで開発された逐次型推論マシンＰＳＩ上の emacs風のテキストエディタ pmacs[5]の機能を利用した汎用構造エディタであり，以下のような特徴を持つ．

①文法に対する汎用性・・・文字列を構文解析し構文木を出力するパーサと構文木を文字列に変換して表示するプリティプリンタが使用する文法を，ユーザが自由に定義できる．文法の定義には BNF風の形式を使用するが，「繰り返し」の表現などは，記述しやすく，それにより規定される構造が直観に合うものとなるように工夫した．

②コマンドの拡張容易性・・・SEMACSのすべてのコマンドが，使用している文法によらずに使用できる．各言語固有のコマンドは，それらの基本的なコマンドに付加して実現する．pmacsのシステム構成の良さとpmacs，SEMACS両者の記述言語ＥＳＰがオブジェクト指向言語であることにより，コマンドの付加が容易になっている．

③ユーザインタフェースの工夫・・・操作性の向上を図るため以下のようなユーザインタフェースの工夫を行った．

- SEMACSでは，文字列を編集する文字列モードと，構文解析の結果である構文木を直接編集する構造モードの２つのモードがあるが，文字列モードではpmacsのほとんどすべてのコマンドが使用できる．また両モード間の行き来もパーサの部分構文解析機能などにより円滑に行うことができる．
- マウスのポインティングにより，編集対象箇所を指定できる．
- 本表示機能を利用することにより，構文木の構造をグラフィカルに見ることができる．また文法案内機能（後述）をこの機能と組み合わせて使用することにより，構文木をこの画面上で展開することができる．

3．文法定義と表示法

3.1　文法定義，構文解析

SEMACSで使用する文法は，利用者が BNF風の形式で自由に定義することができる．以下に例を示す．

```
program ::= list(statement, ";").
statement ::= "if", test, "then", statement,
                              ["else", statement ] |
              "while", test, "do", statement   |
              t(name), ":=", expr              |
              "(", program, ")"                |
              "write", t(name)                 |
              "read", t(name) .
test ::= expr, c-op, expr.
expr ::= expr, op2, expr1 | expr1.
expr1 ::= expr1, op1, expr0   |   expr0.
expr0 ::= "(", expr, ")"   |
          t(name)  |  t(integer).
c-op ::= "="   |   ">"   |   "<".
op2 ::=  "+"   |   "-".
op1 ::=  "*"   |   "/".
```

この定義のうち，" "で囲まれたものとt( )と示されたものは終端記号で，それ以外は非終端記号である．

t(name)は任意の識別子，t(integer)は任意の整数をそれぞれ表している．定義の1行目はSEMACS文法で繰り返しを示すための特殊な表現であるが，「programは";"で区切られたstatement1回以上の繰り返しである」ことを示している．従来の

```
program ::= statement, [";", statement] *
```

（＊は1回以上の繰り返し，［　］は省略可能を表す）あるいは，

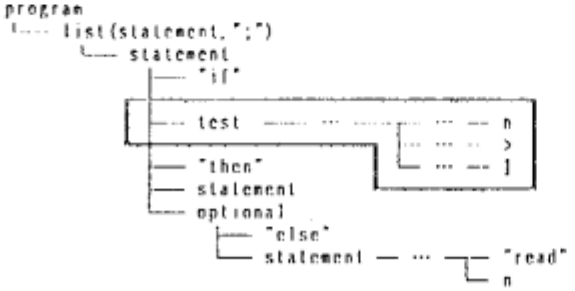
```
program ::= statement, [";", program]
```

といった表現より自然な記述となっていると判断したためである．区切り記号なしの繰り返しは，list(statement)というように定義する．

上記の形式で記述された文法ファイルから，システムは構文解析と文法案内に必要なデータを生成する．構文解析は，左決定木による汎用パーサLDT[10]を使用している．

3.2　表示法，プリティプリント　まず構文木とその表示の例を以下に示す（文法は先の例のもの）．

・構文木



・表示

```
if [ n > ] then $statement ``
           else read n
```

――で囲まれた部分はエリア（編集対象部分を示すもので，テキストエディタのカーソルに相当する）を表すものとする．構造モード時の表示は，構文木の葉の部分のみをプリティプリントすることで実現している．エリアは構文木中の一つの部分木であり，白黒反転して表示される．非終端記号でまだそれ以下の部分木が存在していないもの，およびネロフラスト（後述）されている部分木は，それぞれ，

```
$symbol      $$symbol
```

（symbolは部分木のトップの非終端記号）

というように表示される．またエリアの部分木の根となっている非終端記号（この例ではtest）をエリアのトップカテゴリと呼ぶ．

プリティプリントに必要な，改行・字下げなどの情報も，構文解析に必要な文法情報と同様にユーザが自由に定義できる．

4.　システム構成

SEMACSのシステム構成は図1の通りである．

SEMACSがマンマシンインタフェースの基本として用いているpmacsは

・ウィンドウ（表示系）
・バッファ　（編集対象）
・コマンド　（編集機能）

の各要素を独立して実装している．SEMACSは基本的に上記の各要素に構造編集機能をマージすることで実現されている．その結果pmacsの持つ操作性の良さを充分利用することができ，これはまたSEMACSの一つの特徴ともなっている．以下にその概要を操作性の特徴を中心に述べる．

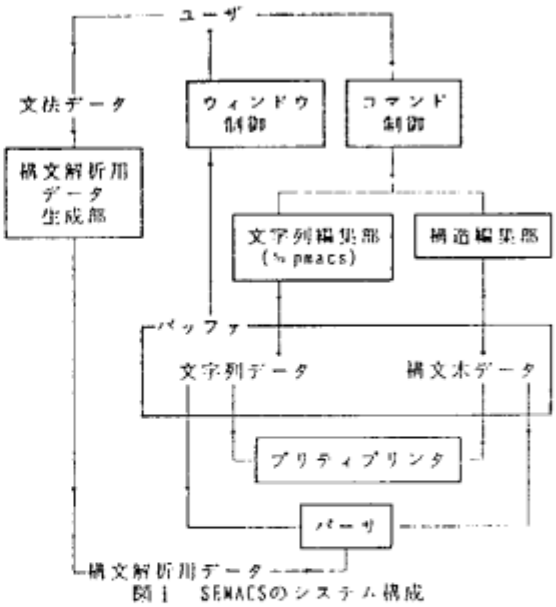


図1　SEMACSのシステム構成

4.1　バッファのオブジェクト化　バッファは，一連の編集操作の対象となる一つの単位である．SEMACSでは，このバッファをオブジェクトとして，編集機能とは独立して保持できる．したがってモジュールの独立性の良さとともに以下のような使用上の利点がある．

①複数のバッファの操作・・・バッファはバッファプールに保持されており，使用者は複数のバッファに対する編集／確認操作を円滑に行える．

②バッファに対する独立した機能の使用・・・編集機能とは別にバッファに対し，バッファ一覧表示，プリント出力，セーブ，ロードなどの機能が使用できる．

③モードの保持・・・文字列モードと構造モードの2つのモードがバッファ及びファイルに保持されている．したがって使用者はバッファを切り換えることで，そのバッファの持つモードで編集を行える．また，逆にバッファの持つモードを切り換えることで，同一バッファに対し文字列編集と構造編集が行える．特に文字列編集機能はpmacsの持つ強力な編集機能のほぼすべてを使用できる．

4.2　ファイル　SEMACSで用いているファイルには，モードを保持しているという特徴がある．これは前述の各モードにおいて編集イメージを保存できることを示す．しかし，使用者はそれぞれのファイルがどのモードか意識しておく必要はなく，SEMACSがファイルの読み込み時に自動判定を行う．また，構造モードのファイルには表示イメージとともに構造情報もセーブされているため，構造モード時の作業再開のための特別な操作（構文解析など）は不要である．これに対し文字列モードのファイルはpmacsと完全に互換性がある．

4.3　コマンドキーアサイン　SEMACSで用いられるコマンドはすべてpmacsのコマンドキーアサイン機能が使用できる．従って基本的には使用者が好みのキーアサインでコマンドを使用できる．ただし構造編集コマンドのキーアサインの既定値は左手だけで入力できるように工夫されている．これは左手がキーボード押下，右手がマウス操作を行うという前提で考えられているからである．

5. 構造編集コマンド

SEMACSには，文字列モードと構造モードの２つがある．ここでは構造モード時に使用できる特徴的なコマンドを説明する．

5.1 モードの切り換え　次の４つのコマンドがある．

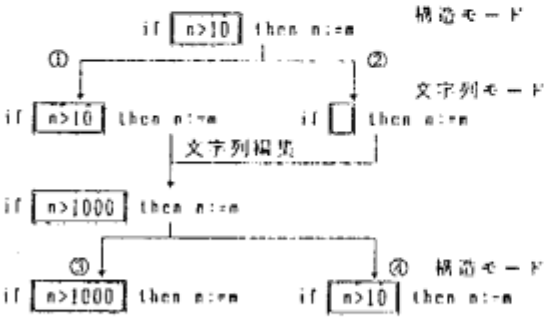
・構造⇒文字列（エリアを文字列編集の対象とする）…①

・構造⇒文字列（エリアの内容を削除）…②

・文字列⇒構造（エリアの内容を構文解析）…③

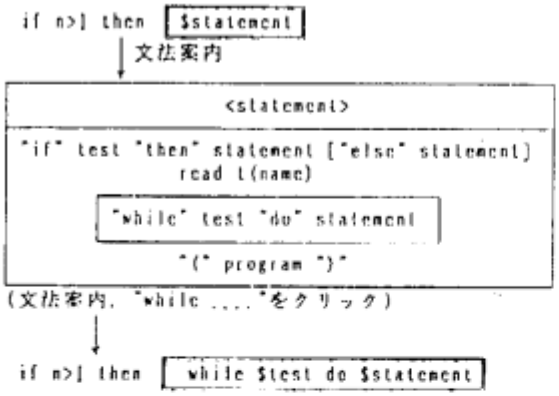
・文字列⇒構造（前回の構造モード時のエリアに復元）…④

例



文字列モード時には，エリア以外の部分もエリア内部と同様に編集することができるので，その部分のテキストを参照したり複写してエリア内部のテキストを編集することができる．ただし，エリア以外の部分の文字列モード時の変更は，再び構造モードに戻るときには無視され，前の構造モードのときのイメージが復元される．

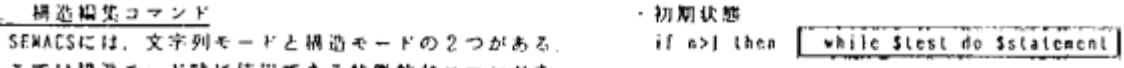
5.2 文法案内　エリアのトップカテゴリの展開の可能性をメニューウインドウの形式で表示し，ユーザの選択により展開を行い，結果を示す機能である．以下に例を示す．



またマウスの別のクリックでは，実際の展開は行わず詳細な情報を見ることができる．上の例では，test，statementに関するウィンドウが順に表示されるが，そのウィンドウ上でいずれかの項目がクリックされたときには再帰的にその項目に関する情報に関する処理を行う．

5.3 エリアの移動　エリアの移動を行うコマンドには，親，子，兄弟など現在の部分木との構文木中の相対的な位置関係を指定するものと，画面中で移動させたい場所を直接マウスにより指定する方法の２通りを用意してある．以下に例を示す．

・初期状態



・コマンドと移動後のエリア

コマンド	移動後のエリア
長子	while
末子	$statement
第２子	$test
長兄	if
兄	then
親	if n>1 then while $test do $statement

マウスポインティングでは，画面上の指定箇所を含む最小の部分木にエリアを移動することができる．また親への移動のコマンドはマウスにもキーアサインされているため，マウスのみを用いて任意の部分木にエリアを移動することができる．

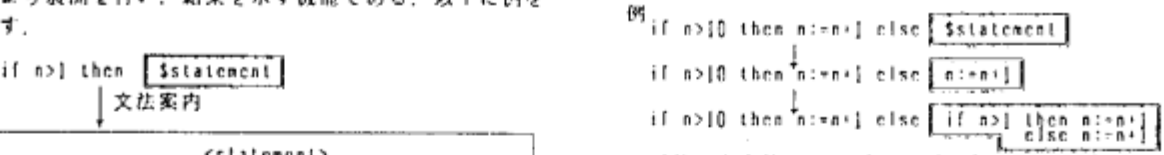
構文木を直接取り扱う場合，

expr--expr1--expr0--t(name)--X

といった一本道の木が随所に現れてしまう可能性があるが，SEMACSでは「兄弟のないノードはエリアのルートになれない」という制約を設けている．（上の例で expr1 や expr0はエリアのルートになれない）この制約により，エリア移動のコマンドが入力された場合（それが行き場のない移動でない限りは）表示上のエリアも必ず変化する．

5.4 探索・置換　各種のコマンドが用意されているが，ここでは「現在のエリアのトップカテゴリと同じトップカテゴリを構文木の前方に探索し，エリアをその内容に置き換える」という処理を行うコマンドを，連続して用いた例を示す．

例



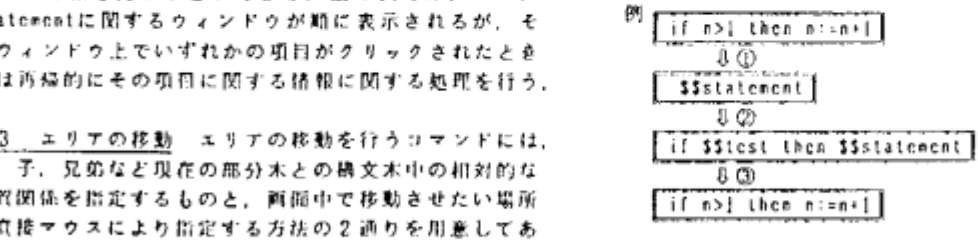
5.5 ホロフラスティング　詳細な情報を隠蔽して省略された表示をしたり，逆にその隠蔽を解除して表示したりする機能である．SEMACSではホロフラスティングは部分木単位で行っており，ホロフラスティングされている部分木は，そのトップカテゴリが"$$"つきで表示される．削除の場合と異なり，表示が省略されるだけでその内容は保持されている．以下のコマンドがある．

・エリア全体のホロフラスティング…①

・エリア内のホロフラスティングのレベルを一段下げる…②

・エリア内の全ホロフラスティングを解除する…③

例



6. 木表示

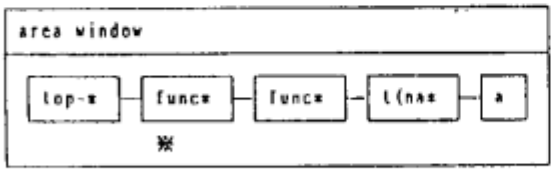
SEMACSの木表示モジュールはICOTで開発されているエキスパートシステム開発支援ツールProlon[1]の木表示モジュールをもとに，SEMACS用の機能を付け加えたものである．主な機能は，

(1)エリアの木表示

(2)ユーザが選択したノードをルートとする部分木の表示

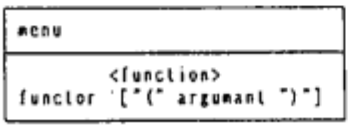
(3)文法案内

である．これらのうち，特に重要なのが文法案内である．まず，文法案内の例を下図に示す．（i）の木表示ウィンドウは構造モード時にあるコマンドを入力することにより，テキスト表示のウィンドウとは別に生成される．（i）の※のノード上であるマウス・クリックを入力すると，（ii）に示す構文ガイダンスウィンドウが表示される．メニュー項目をあるマウス・クリックで選択すると，※のノードが木表示ウィンドウ上で（iii）のように展開される．



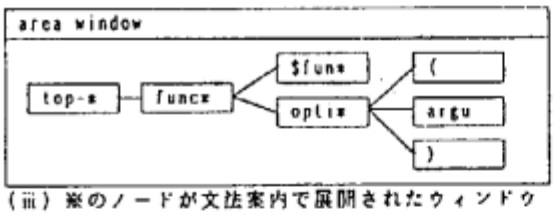
（i）もとの木表示ウィンドウ

⇩ 文法案内指示



（ii）文法案内ウィンドウ

⇩ 展開指示



（iii）※のノードが文法案内で展開されたウィンドウ

木表示ウィンドウ上で展開が行われてもテキスト表示のウィンドウとは同期をとらず，木表示ウインドウ上での操作を終える時にその結果をテキスト表示のウィンドウに反映するかしないかをユーザが選択するようになっている．こうすることによりテキスト表示の方は変えないままで，木表示ウィンドウ上で構造上のどの位置かを常に明確に把握しながら文法案内の展開を行って行くことができるわけである． もちろん，この木表示モジュールは，構造をグラフィカルにわかりやすく認識するという点だけでも役に立つものである．今後の拡張としては，

(1)テキスト表示ウィンドウと同等の，構造の複写・削除・探索などの編集機能をグラフィカルな木の形で実現すること

(2)木全体を概略的に示すウィンドウの実装，結合

などが考えられる．

7. 応用例

7.1 CAPでの応用 CAP(Computer Aided Proof) システムは，人間の証明の記述，編集，検査といった活動を支援するシステムである．そこでSEMACSは，ユーザとシステムとのインタフェースの役割を果たす．使用する文法は同プロジェクトで研究開発された証明記述用の言語PDL(Proof Description Language)である．SEMACSの通常の機能に加えて，CAPシステムでは，証明検査箇所のリアルタイム表示，推論規則を利用した証明の自動展開などのコマンドを付加して使用している．証明の自動展開機能で使用する置き換え規則は，PDLの文法に依存していない機能なので他の文法でも使用することができる．現在 CAPシステムは，証明記述がすべて終了した時点でその内容の妥当性を検査するバッチ的な処理を行っているが，SEMACSの機能を核として対話的な検査を行うシステムへの拡張を予定している．

7.2 Kappaでの応用[5] Kappa(Knowledge Application Oriented Advanced DBMS/KBMS)は知識データベースシステムである．この中でSEMACSは主にメタデータの作成／修正を行うメンテナンスツールに組み込まれて使用されている．ここでは非正規関係のメタデータを扱う関係上，複雑な構造を持ったスキーマ定義情報などを正しく作成／修正することを要求され，通常のテキストエディタでは操作性や正当性に問題が出てくる．このためSEMACSをベースにしたメンテナンス・ツールは，構造編集や文法案内の機能によって，利用者への強力な支援ツールとなっている．さらに，意味制約からスキーマの自動設計ツールやデータベースアクセス用のコマンドを追加して設計支援システムとしての拡張を予定している．

謝辞

pmacs関係でご助言いただいたICOT第4研究室の佐藤裕幸氏と，Proton関係でお世話になった開発担当者の皆様に感謝致します．

文献

[1] Bertrand, M. : CEPARGE:Towards Computer-Aided Design of Software, Interactive Software Engineering, Inc., 1987.

[2] 原田賢一：構造エディタ，共立出版，1987.

[3] ICOT: SIMPOS文書編集機能説明書，1987.

[4] ICOT: SEMACS使用説明書，1988.

[5] 河村，他： 知識ベース管理システムKappa —利用者インタフェース—，第35回情処全国大会論文集，pp.1615-1616，1987.

[6] 溝口徹夫，横田一正，内田俊一： 知識ベース管理システムKappa，第34回情処全国大会論文集，pp.1603-1604，1987.

[7] Sakai, K.: Toward Mechanization of Mathmatics - Proof Checker and Term Rewriting System —, ICOT Technical Report TR-348, 1988.

[8] 坂井公： DCGのボトムアップ型構文解析アルゴリズムの新方式について，Proceedings of The Logic Programming Conference'86, pp.13-18, 1986.

[9] 澤本他：ＰＳＩ上のエキスパート開発支援ツール(1)～(4)，第33回情処全国大会論文集，pp.1115-1122.1986.

[10] 横田一正：知識ベース基本ソフトウェアKappa，第５回５Ｇシンポジウム，June, 1987.