プロダクションメモリのモジュール構造化機能をもつ適応プロダクションシステム†

桃内佳雄†† 小林 茂††† 宮本衛市††

基本的なプロダクションシステムにおいては，プロダクションメモリは一様な構造をもつプロダクションルールの集合として構成され，これはプロダクションシステムにおけるプログラミングのしにくさ，効率の悪さなどの一つの原因となっている．また，プロダクションシステムは学習や問題解決のモデルの作成に広く応用されているが，一様な構造をもつプロダクションメモリでは学習や問題解決などのための構造的な知識を適切に表現することができない．さらに，学習や問題解決などのモデルの作成のためには，従来の適応プロダクションシステムがもつプロダクションルールの生成・付加機能に加えて，プロダクションメモリへの柔軟な動的アクセス機能の装備が必要である．本論文では，おもな問題領域を学習や問題解決のモデルの作成とする適応プロダクションシステム APSH について述べる．APSH のおもな特徴は，プロダクションメモリのモジュール構造化機能をもつこと，プロダクションルールの付加に加えてプロダクションメモリおよびワーキングメモリへの柔軟な動的アクセス機能と情報付与機能をもつこと，プログラムの作成・編集を支援するエディタやトレーサを内蔵し，柔軟なコマンド言語をもつ対話型プログラミングシステムであることなどである．プロダクションメモリのモジュール構造化は，プログラムの作成・修正・理解，そして学習や問題解決などのための知識の表現を容易にする．

1. まえがき

人工知能あるいは認知心理学の分野において，人間が行う知的作業を実現するシステムや学習とか問題解決などの認知過程のモデルの作成のために，プロダクションシステム (Production System, PS) は広く応用されている．PS の初期の応用の一つとして，Newell が提案した基本的 PS の構成は，問題領域に依存しないもので[1]，いろいろな分野の多くの研究者によって用いられている．しかし，知識工学における PS の応用などにみられるように[2]，より問題領域に依存した構成をもつ PS も多く，その有用性も確かめられている．また，PS は，学習や問題解決などのモデルを計算機上で実現するためのプログラミング言語，およびその処理系として位置づけることもでき，プログラミングシステムとしての PS についての考察もいくつか行われている[3),4]．本論文では，学習や問題解決のモデルを作成するためのプログラミングシステムとしてのプロダクションシステム APSH (Adaptive Production System—Hokkaido university) を提案し，その構成と機能について述べる．

一般に，基本的 PS は次のような構成要素をもつ[1),5]．

(PS 1) プロダクションメモリ (Production Memory, PM)：条件部と行為部の対から構成されるプロダクションルール (Production Rule, PR) の集合を格納する．

(PS 2) ワーキングメモリ (Working Memory, WM)：データ要素 (Data Element, DE) の集合（データ）を格納する．

(PS 3) インタープリタ (Interpreter, IP)：照合，競合解消，および行為実行の三つの動作の繰返しを行う．PS におけるプログラムは，PM の PR 集合と WM の DE 集合とから構成され，IP はプログラムの解釈実行を行うものと考えることができる．

PS は学習や問題解決のモデルを作成するために都合のよい，いくつかの優れた特徴をもつことが知られているが[4),6]，しかし一方，PM の PR 集合全体を一つの単位として構成する基本的 PS の方式では，プログラムの構造化などの困難さによるプログラミングのしにくさ，およびプログラム実行上の効率の悪さなどの欠点を生じ，これらの欠点の克服も含めて，PS の構成に関する多くの研究が進められている[4)~11]．

Lenat と McDermott[5] は，基本的 PS の拡張の方法とその利点について系統的な考察を行い，PS を適用する問題領域に依存した，より適切な PS の構成の

† An Adaptive Production System with Module-Structured Production Memory by YOSHIO MOMOUCHI (Division of Information Engineering, Graduate School of Engineering, Hokkaido University), SHIGERU KOBAYASHI (Toshiba Corporation) and EIICHI MIYAMOTO (Division of Information Engineering, Graduate School of Engineering, Hokkaido University).

†† 北海道大学大学院工学研究科情報工学専攻
††† 東京芝浦電気(株)

可能性のいくつかを示した．その一部を要約すると，

(LM 1) 複数個の PM をもつこと，

(LM 2) PR をデータとして取り扱うこと，

(LM 3) PR の複雑な構成を許すこと，

などである．

Waterman の適応プロダクションシステム PAS-Ⅱ[10] は，PR をデータとして取り扱うことにより，プログラム中の行為によって PM への PR の付加を行うことを可能にしている．Waterman は PAS-Ⅱ 上で，いくつかの適応学習のプログラムを作成し，WM だけでなく，PM への動的なアクセス機能の有用性を示している．

本論文で述べるプロダクションシステム APSH の設計に当たっては，Lenat と McDermott によって論じられた可能性のなかで，とくに (LM 1) と (LM 2) に注目して基本的な設計方針を設定した．設計方針ならびにそれに伴うシステムのおもな機能を次のようにまとめることができる．

(1) PS におけるプログラムは PR 集合と DE 集合とから構成され，したがって，プログラミングの仕事は，与えられた問題を解決するための PR 集合と初期 DE 集合とを作成することになる．前述したように，プログラミングのしにくさ，効率の悪さなどの基本的 PS の欠点は，PM の PR 集合全体を一つの単位として作成するということに主として起因するものである．APSH では，この欠点を克服するために，第1に，ソフトウェア工学における，よいプログラムを作成するための方法として提案されているモジュラプログラミングの有効性を考慮し[13]，第2に，認知プロセスのモデルを作成するためのプログラミングシステムとして，問題解決における階層的な目標構造の表現のしやすさ，より一般的には，認知プロセスにおける構造化された手続きの表現のしやすさということを考慮して[12]，PM のモジュール構造化という方式を採用する．これは複数個の PM というよりは，1個の PM を構造化したものとして把えることができる．PM の構成要素であるモジュールは，PR モジュールと呼ばれ，PR の順序集合として構成される．

(2) Waterman による PAS-Ⅱ[10] の構成の方向に沿って，学習・適応機能を強化する．PR の生成および付加だけでなく，既存の PR を参照しそれを削除したり，一部修正したりすることができる．また，PR と DE に重みやフラグなどによる情報を付与し，それらを利用することによる有用な学習モデルの作成を助ける．

(3) PS を対話型プログラミングシステムとして構成する．PR の行為部の構成要素である行為をコマンドとして用いることができ，柔軟な対話型プログラミングが可能である．さらに，エディタとトレーサを内蔵し，プログラミングを支援する．

2. プログラムの構造

APSH におけるプログラムは PM の PR モジュール群とそれらが共有する WM のデータとから構成される．モジュールには，プログラマがエディタを用いて PM に作成する PR モジュールと，あらかじめシステムに組み込まれている，固有の名前と機能をもつ，50個の基本モジュールがある．次に，具体的なプログラム例を通してプログラムの構造および構造化について説明し，最後に，PM のモジュール構造化の利点についてまとめる．

2.1 階乗計算プログラム

図1は，階乗計算を行うとともに，階乗計算規則に対応する新しい PR の生成と付加を同時に行ってゆくプログラムである．このプログラムは，F, FACT, MAKERULE, MULTSOL, および PRINT という名前の5個の PR モジュールから構成される．

PR モジュールは，モジュールの名前と必要個数の

```
(F)
(1 (CUL 0) => (DEP (SOL 1)) (RETURN))
(2 (CUL ?C) => (ADD ?C -1 ?A) (REP #1 ?C ?A) (F) (MULTSOL ?C)
  (MAKERULE ?C) (RETURN))

((FACT ?1) = (CW) (DEP (CUL ?1)) (F) (PRINT))

(MAKERULE ?M)
(1 (SOL ?1) => (COND (CUL ?M)) (ACTION (DEP (SOL ?1)) (RETURN)) (PROD F)
  (RETURN))

(MULTSOL ?M)
(1 (SOL ?1) => (MULT ?1 ?M ?S) (REP #1 ?1 ?S) (RETURN))

(PRINT)
(1 (SOL ?1) => (SAY KOTAE= ?1) (RETURN))
```

図1 階乗計算プログラム
Fig. 1 Factorial program.

パラメータの並びとから構成される一つのリスト（モジュール頭部）と，リストによって表現されるPRの並び(モジュール本体)で構成される．たとえば，Fでは，(F)がモジュール頭部で，その下の二つのリストの並びがモジュール本体である．FACT, MAKERULE, および MULTSOL はパラメータをもち，仮パラメータが，それぞれ変数 ?1, ?M, ?M により表されている．これらの変数は？変数と呼ばれる．

PR は，PR の名前，条件部，⇒，および行為部から構成される．たとえば，Fのモジュール本体の PR,

(1 (CUL 0)⇒(DEP (SOL 1)) (RETURN))

において，第1要素“1”は PR の名前（名前は正整数），名前を除いた⇒の左辺が条件部，そして右辺が行為部である．条件部，行為部ともにリストの並びであり，それぞれの構成リストは条件要素 (Condition Element, CE), 行為と呼ばれ，上の PR では，条件部と行為部は，それぞれ一つの CE，二つの行為から構成される．⇒の左辺に“−”が含まれている場合は，その左側にある条件要素を正の条件要素，右側にある条件要素を負の条件要素と呼ぶ．

FACT の記法は，下に示す PR モジュール表現に対する簡略記法である．ただし，簡略記法を用いた場合，＝の左側がモジュール頭部，右側が無条件で実行される行為の列を表す．

(FACT ?1)
(1 ⇒(CW) (DEP(CUL ?1))
(F) (PRINT)(RETURN))

PR モジュールから PR モジュールを呼び出すこともでき，その呼出しは行為部の行為として表現され，たとえば，Fの PR-2 の行為 (F) はF自身の再帰的呼出しを，行為 (MULTSOL ?C) と (MAKERULE ?C) は，それぞれ MULTSOL と MAKERULE のパラメータ付き呼出しを表している．また，(ADD ?C -1 ?A),(REP #1 ?C ?A), および (RETURN) などは，基本モジュールの呼出しを表している．DEP, CW, COND, ACTION, PROD, MULT, SAY なども基本モジュールである．このプログラムの PR モジュール間の呼出し関係による，PR モジュール関連構造を図2に示す．このように，APSH におけるプログラムにおいては，PR モジュール間の呼出し関係による PR モジュール関連構造は，ある特定の一つの PR モジュールをトップとする階層構造を構成することとなる．

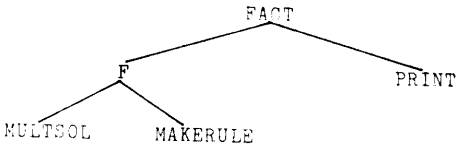

図 2 階乗計算プログラムのモジュール関連構造
Fig. 2 Hierarchical structure of modules of factorial program.

図2は，HIPO で用いられている機能階層図に対応しており[13]，プログラムの機能を，まず階乗計算 (F) と結果出力 (PRINT) に分割し，さらに階乗計算を積計算 (MULTSOL) と規則生成・付加 (MAKERULE) に分割して機能階層図を作成し，各機能に対応してモ

```
(SERIATE)
(1 (STATE START) (POOL ?1) => (TRANSFER))
(2 (STATE START) (POOL) => (RETURN))

(TRANSFER)
(1 (STATE (GET ?1)) (POOL) => (ERASE #2 ?1) (REM #1)
   (DEP (STATE (HAVE ?1))) (PLACE))
(2 (STATE (PERCEIVE ?1)) => (REM #1) (PROPDEC ?1))
(3 (STATE GOOD) => (REM #1) (DEP (STATE START)) (RETURN))
(4 (STATE START) => (REM #1) (PROGETB))

(PLACE)
(1 (STATE (PUT (?1 ?2))) => (REM #1) (PROADDB ?1 ?2) (RETURN))
(2 (STATE (HAVE ?1)) => (REM #1) (PROPUTB ?1))


(PROADDB ?X ?Y)
(1 (= ?X END) (LINE) => (APPEND #1 ?Y) (DEP (STATE (PERCEIVE ?Y)))
   (RETURN))

(PROCHEC ?X)
(1 (LLIST ?1 ?2) (< ?1 ?2) (= ?1 ?X) => (DEP (STATE GOOD)) (REM #1)
   (RETURN))
(2 (LLIST ?1) => (DEP (STATE GOOD)) (REM #1) (RETURN))

(PROCOMP)
(1 (BIG ?1) (PBLOCK ?2) (< ?2 ?1) => (REM #2))
(2 (BIG ?1) (PBLOCK ?2) (< ?1 ?2) => (REM #1) (DEP (BIG ?2)) (REM #2))
(3 (BIG ?1) => (REM #1) (DEP (STATE (GET ?1))) (RETURN))

(PROGETB)
(1 (POOL ?1) (PSAVE) => (DEP (PBLOCK ?1)) (ERASE #1 ?1))
(2 (POOL) (PSAVE ?1) => (REM #1) (DEP (BIG 0)) (PROCOMP) (DEP ?1)
   (REM #2) (RETURN))
(3 (POOL) => (DEP (PSAVE #1)))

(PROLIST)
(1 (LINE ?1) => (DEP (LBLOCK ?1)) (ERASE #1 ?1))
(2 (LBLOCK ?1) (LLIST) => (APPEND #2 ?1) (REM #1))
(3 (LINE) => (RETURN))

(PROPDEC)
(1 (LINE ?1) (LSAVE) => (DEP (LLIST)) (PROLIST) (ERASE #1 ?1))
(2 (LINE) (LSAVE ?1) => (REM #1) (PROCHEC ?X) (DEP ?1) (REM #2)
   (RETURN))
(3 (LINE) => (DEP (LSAVE #1)))

(PROPUTB ?X)
(1 => (DEP (STATE (PUT (END ?X)))) (RETURN))
```

図 3 長さの系列化プログラム
Fig. 3 Seriation program.

ジュールを作成するというプログラミングを考えることができる．このようなモジュール単位のプログラム作成は，プログラムの作成，修正，および理解を容易にする．このプログラムをトップモジュール FACT から，P-FACT と呼ぶことにする．

WM のデータは，データ要素(DE)と呼ばれるリストの並びとして表現される．データの例を次に示す．

(CUL 0) (SOL 1)

(CUL 0) と (SOL 1) が DE であり，これは，P-FACT により 0! を計算したときに WM に置かれるデータである．

2.2 長さの系列化プログラム

図3は，長さの系列化プロセスモデルのプログラムである．このプログラムでは，長さの異なる複数個の棒の山 (POOL) のなかから，順次最も長い棒を取り出し，長さの順に列 (LINE) の右端にそれを置いてゆくという方略をとる．

安西らは[12]，長さの系列化プロセスにおいては，問題解決の目標構造の導入によってプロセスの説明が容易になることを示している．彼らのモデルでは，長さの系列化プロセスにおける目標として，SERIATE (棒を長い順に並べる)，TRANSFER (棒を山から列に移す)，および PLACE (棒を列に加える) という三つの目標を設定し，問題を最終的に解決するまで，SERIATE-TRANSFER-PLACE という目標の階層構造の下で，どのような手続きを実行すべきかの制御を行っている．

図3のプログラムでは，これらの SERIATE, TRANSFER, および PLACE という目標下での仕事を，それぞれ，SERIATE, TRANSFER, および PLACE という名前の PR モジュールにまとめている．PROADDB から PROPUTB までのモジュールは，これらの目標下で実行を制御される，棒を具体的に操作するための手続き群である．モジュール間の関連構造を**図4**に示す．これは，目標構造と手続きの間の階層的な関連構造を明らかに示すものとなっている．すなわち，問題解決における階層的な目標構造を，目標とモジュールを対応させることにより，モジュール間関連構造として容易に実現することができる．目標および手続きがモジュール単位でまとめられるので，プログラムの作成，修正，そして理解も容易となるであろう．このプログラムをトップモジュール SERIATE から，P-SERIATE と呼ぶことにする．

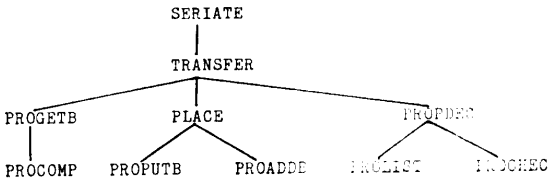

図4 長さの系列化プログラムのモジュール関連構造
Fig. 4 Hierarchical structure of modules of seriation program.

比較のために，図3の，目標と対応する三つのモジュール SERIATE, TRANSFER, および PLACE の仕事をモジュール構造化しないで作成したプログラムを**図5**に示す．これは，安西らが作成したプログラム[12]とほぼ対応している．これでは，第1要素に GOAL というアトムをもつ条件要素の形で目標が表現され，WM の中に目標に対応する DE を置き，それを参照しながら目標の階層構造を制御しなければならない．目標の階層構造とその制御が PR の集合と WM の中に埋め込まれた形となり，このような方式では，PR の数が多くなってきて，階層構造が複雑になってくると，プログラムの作成と理解，とくに構造の理解，が困難になってくると考えられる．

2.3 モジュール構造化の利点

2.2 節での具体例による検討を通して，PM をモジュール構造化することによる利点を次のようにまとめることができる．

(M1) プログラムをモジュール単位で作成することができるので，プログラムの作成，修正，および理解が容易になる．

(M2) 学習や問題解決のプロセスに含まれる目標構造などの階層的な構造に対応して，機能単位のモジュール群によりプログラムを階層的に構成することによって，モデルのよりよい記述が可能となる．

さらに，次章で述べる，プログラムの実行と関連して，次のような利点をあげることができる．

```
(1 (GOAL PLACE) (STATE (PUT (?1 ?2))) => (REM #2) (PROADDB ?1 ?2)
  (REM #1))
(2 (GOAL PLACE) (STATE (HAVE ?1)) => (REM #2) (PROPUTB ?1))
(3 (GOAL TRANSFER) (STATE (GET ?1)) (POOL) => (ERASE #2 ?1) (REM #2)
  (DEP (STATE (HAVE ?1))) (REM #1) (DEP (GOAL PLACE)))
(4 (GOAL TRANSFER) (STATE (PERCEIVE ?1)) => (REM #2) (PROPDEC ?1))
(5 (GOAL TRANSFER) (STATE GOOD) => (REM #2) (DEP (STATE START))
  (REM #1)
(6 (GOAL TRANSFER) (STATE START) => (REM #2) (PROGETB))
(7 (GOAL SERIATE) (STATE START) (POOL ?1) => (REM #1)
  (DEP (GOAL TRANSFER)))
(8 (GOAL SERIATE) (STATE START) (POOL) =>(RETURN))
```

図5 モジュール構造化しない長さの系列化プログラム
Fig. 5 Seriation program without module-structuring.

(M 3) プログラムの実行において，起動する PR の選択をモジュール単位で行うので，効率の向上を期待することができる．

3. プログラムの実行

3.1 プログラムの起動と終了

プログラムの実行は，プログラムのトップモジュール名をコマンド（パラメータをもつ場合には実パラメータ付きコマンド）として入力し，その PR モジュールを起動することにより開始する．PR モジュールから PR モジュールの呼出しは PR の行為部に含まれる行為として行われる．呼び出された PR モジュールで基本モジュール RETURN を実行すると，それを呼び出した PR モジュールへ戻る．したがって，トップモジュールでの RETURN の実行がプログラムの実行終了となる．

3.2 PR モジュールの実行

PR モジュールは PR の集合であり，PR モジュールの実行は基本的 PS の実行に対応し，照合，競合解消，および行為実行の三つの過程の繰返しである．次に APSH におけるそれぞれの過程について説明する．

① 照合：PR の左辺の CE に WM の DE が適合するとは，CE のすべての要素が DE の要素として同じ順序で含まれていることである．CE と DE の照合において，CE 中の付値されていない？変数には任意のアトムまたはリストを付値することができる．以上のことを考慮にいれて，PR の左辺が WM のデータによって満たされるための条件を次のように述べることができる．「すべての正の CE に適合する DE が存在し，かつ，すべての負の CE に適合する DE が存在しない．ただし，一つの DE は一つの CE に対してしか適合することができない」．照合は後戻りを伴って処理される．

② 競合解消：APSH における競合解消の規則は，「照合の過程で条件部がデータにより満たされる PR の中，モジュール頭部に最も近い PR を選択する」というもので，これはマルコフ型の制御構造に対応する．このように，基本的には，順序に基づくものであるが，PR には 0 から 1 までの小数点以下 3 桁までの重みを付与することができ，PR はモジュール頭部から重みの大きい順序で並べられる機構になっているので，上の競合解消の規則は，「最も大きな重みをもつ PR を選択する」と言い換えることができる．また，PR の重みを基本モジュールによる行為によってプログラムの中で動的に変更することができ，それがこの競合解消の規則に柔軟性を与えている．

③ 行為実行：起動すべき PR が選択されると，その右辺の行為が左から順番に実行される．すべての行為を実行し終えると PR の行為実行は終了する．

3.3 実 行 例

（1） P-FACT の実行例

図 6 が P-FACT の実行例である．〈ATTEND〉により表示されるコマンド入力待ち状態で，"FACT 5;" を入力し，"KOTAE=120" を得ている．新しい PR の生成も表示されている．"LIST F;" はモジュール F のリスト出力の指示である．次に，5! の計算過程をもう少し詳しくみてみよう．P-FACT において 5! を計算するために，コマンド "FACT 5;" を入力し，FACT を起動すると，(CW) によって WM をクリアする基本モジュール CW，(DEP (CUL ?1)) によって WM にデータを書き込む基本モジュール DEP，引き続いて PR モジュール F，PRINT が順番に起動される．FACT を呼び出した時点で，変数 ?1 には '5' が付値され，DEP モジュールにより WM に (CUL 5) が書き込まれる．F が起動されると，WM のデータに応じて PR-1 か PR-2 が選択されることになるが，WM の (CUL 5) が PR-2 の条件要素

```
< ATTEND >
FACT 5;

PR   3 WAS CREATED.
     (3 (CUL 1) => (DEP (SOL 1)) (RETURN))
PR   4 WAS CREATED.
     (4 (CUL 2) => (DEP (SOL 2)) (RETURN))
PR   5 WAS CREATED.
     (5 (CUL 3) => (DEP (SOL 6)) (RETURN))
PR   6 WAS CREATED.
     (6 (CUL 4) => (DEP (SOL 24)) (RETURN))
PR   7 WAS CREATED.
     (7 (CUL 5) => (DEP (SOL 120)) (RETURN))
KOTAE= 120
< ATTEND >
LIST F;

     (F)
     (7 (CUL 5) => (DEP (SOL 120)) (RETURN))
     (6 (CUL 4) => (DEP (SOL 24)) (RETURN))
     (5 (CUL 3) => (DEP (SOL 6)) (RETURN))
     (4 (CUL 2) => (DEP (SOL 2)) (RETURN))
     (3 (CUL 1) => (DEP (SOL 1)) (RETURN))
     (1 (CUL 0) => (DEP (SOL 1)) (RETURN))
     (2 (CUL ?C) => (ADD ?C -1 ?A) (REP #1 ?C ?A) (F) (MULTSOL ?C)
       (MAKERULE ?C) (RETURN))

< ATTEND >
FACT 4;

KOTAE= 24
< ATTEND >
```

図 6 階乗計算プログラムの実行例
Fig. 6 Execution of factorial program.

(CUL ?C) と適合し，PR-2 が選択され，?C に '5' が付値され，PR-2 の行為部の行為が左から順番に実行されることになる．(ADD ?C -1 ?A) により，?C の値 '5' に −1 が加えられ，その値 '4' が ?A に付値される．(REP #1 ?C ?A) の“#1”は，#変数とよばれ，#変数は #i(i: 正整数)という形で用いられ，条件部の第 i 番目の CE と適合する WM の DE をその値としてとる．この例では #1 であるから，CE (CUL ?C) と適合する DE (CUL 5) をその値としてとる．この基本モジュール DEP による行為は，?C の値，すなわち DE (CUL 5) の '5' を ?A の値である '4' で置き換え，WM には (CUL 4) が残る．以下，再帰的に F を起動し，引き続いて MULTSOL, MAKERULE を起動する．MULTSOL では，F の再帰的呼出しの戻りの結果として 4! の値が求められているので，それに 5 を掛けて 5! を計算し，MAKERULE では「5! は 120」という規則を生成して F に付加する．最後に RETURN を実行して FACT へ戻る．呼び出され，実行されるモジュール，選択される PR の軌跡を示したものが**図 7** である．ただし，MULTSOL と MAKERULE の軌跡の一部の詳細を省略している．

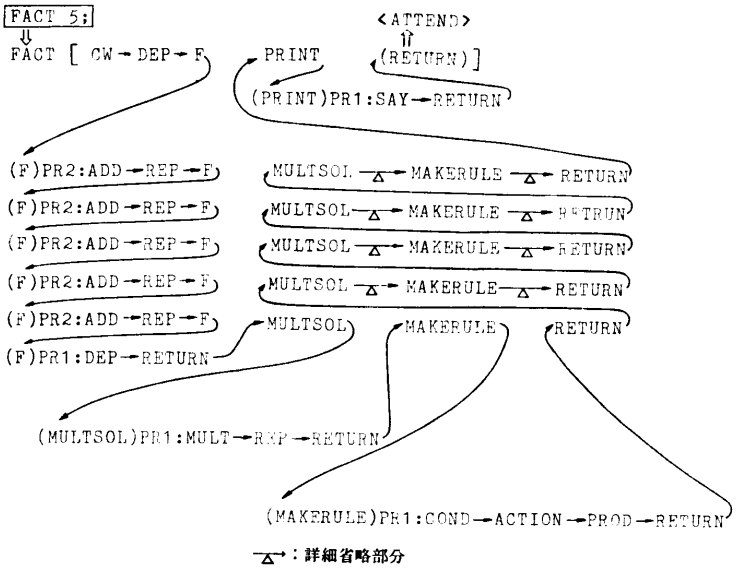

図 7 5! の計算の軌跡
Fig. 7 Trace of the calculation of 5!.

```
< ATTEND >
SERIATE;

     (STATE START) (LINE) (POOL 3 1 4 7 6 2 5)
SERIATE
PR   1    FIRE
TRANSFER
PR   4    FIRE
PROGET8
PR   3    FIRE
     (PSAVE (POOL 3 1 4 7 6 2 5)) (LINE) (POOL 3 1 4 7 6 2 5)
PR   1    FIRE
     (PBLOCK 3) (PSAVE (POOL 3 1 4 7 6 2 5)) (LINE) (POOL 1 4 7 6 2 5)
PR   1    FIRE
     (PBLOCK 1) (PBLOCK 3) (PSAVE (POOL 3 1 4 7 6 2 5)) (LINE)
     (POOL 4 7 6 2 5)
PR   1    FIRE
     (PBLOCK 4) (PBLOCK 1) (PBLOCK 3) (PSAVE (POOL 3 1 4 7 6 2 5)) (LINE)
     (POOL 7 6 2 5)
PR   1    FIRE
                    ( )                    ( )
PR   2    FIRE
     (LLIST 1 2 3 4 5 6 7) (LSAVE (LINE 7 6 5 4 3 2 1)) (POOL) (LINE)
PR   3    FIRE
---  RETURN  ---
     (LLIST 1 2 3 4 5 6 7) (LSAVE (LINE 7 6 5 4 3 2 1)) (POOL) (LINE)
     (LLIST 1 2 3 4 5 6 7) (LSAVE (LINE 7 6 5 4 3 2 1)) (POOL) (LINE)
PR   2    FIRE
PROCHEC
PR   1    FIRE
---  RETURN  ---
     (STATE GOOD) (LSAVE (LINE 7 6 5 4 3 2 1)) (POOL)
---  RETURN  ---
     (LINE 7 6 5 4 3 2 1) (STATE GOOD) (POOL)
     (LINE 7 6 5 4 3 2 1) (STATE GOOD) (POOL)
PR   3    FIRE
---  RETURN  ---
     (STATE START) (LINE 7 6 5 4 3 2 1) (POOL)
     (STATE START) (LINE 7 6 5 4 3 2 1) (POOL)
PR   2    FIRE
---  RETURN  ---
     (STATE START) (LINE 7 6 5 4 3 2 1) (POOL)
```

図 8 長さの系列化プログラムの実行例
Fig. 8 Execution of seriation program.

前章の利点 (M 3) において述べたように，起動すべき PR の選択はモジュール単位で行われる．たとえば，P-FACT の F が呼び出されて実行されるときには，F の二つの PR のなかから起動すべき PR を選択することになり，他のモジュールの PR については選択処理の対象外となる．また，モジュール間結合としては，たとえば，F と MULTSOL 間のようなパラメータを渡すことによる一方向的なパラメータ結合と，たとえば，F と PRINT 間のような WM のデータを共有する共有結合が可能である．F と MULTSOL は共有結合の関係にもある．

(2) P-SERIATE の実行例

図 8 が P-SERIATE の実行例である．長さが 1 から 7 までの 7 本の棒の系列化を実行している．一部省略しているが，トレース情報も出力しており，どのモ

ジュールのどの PR が起動され，WM がどのように変化していくかを知ることができる．これと対応して問題解決の目標の変化とそれに伴う手続きの実行の流れを知ることができる．

4. 学習・適応機能

APSH は，PM や WM への柔軟な動的アクセス機能および情報付与機能を実現することによって，学習モデルのプログラム作成のための基本的な機能を準備している．そのほとんどは組込みの基本モジュールによって実現されているが，そのいくつかのものについて以下で概略を説明する．

4.1 PR の生成・付加・削除

行為 (PROD *a*) は，WM 中の，第1要素が COND, NCOND, ACTION である DE からそれぞれの第2要素を正の CE，負の CE，行為とする PR を生成し，その PR を PR モジュール *a* に付加する．行為 (CHECK *a l*) は，PR モジュール *a* の *l* という右辺をもつ PR を処理の対象として指定する．行為 (REMRULE) により指定した PR を削除することができる．

図6の P-FACT の実行例で，"FACT 5;" の実行によって，PR モジュール F に新しい PR が生成・付加されていることが実行後のプログラムリストからわかる．これは，階乗計算規則を記憶する簡単な暗記学習のプログラムということができる．"FACT 5;" に引き続いて，"FACT 4;" を入力すると，F の PR-6 が起動されて，ただちに "KOTAE=24" が出力されている．

4.2 PR と DE に付与される重みとフラグ

PR と DE には，それぞれ0以上1以下の重みを付与することができる．

MYCIN においても，PR とその条件要素に，それぞれ減衰係数，確実性係数と呼ばれる重みを付与して，PR による推論の結論に対する確実性の計算に用いており，推論過程を AND/OR 木で表したときの AND 節，OR 節それぞれに対して，定められた計算式に従って結論の確実性係数を計算する[14]．

APSH においては，重みの付与ばかりでなく，重みの変更をプログラムによって行うことがきる．PR の重みの変更は，CHECK で指定した PR に対して，(RELUP *n*) によりその重みを *n* だけ増加させることにより行うことができる．DE の重みの変更は，(ACTIVATE *a*) により次の計算式に従って行う．ただし，*a* は重みを変更する DE，*w* は変更後の重みである．

$$w = w_0 + (1 - w_0) \times s \times r$$

w_0 は変更前の *a* の重み，*r* はこの (ACTIVATE *a*) という行為を行為部に含む PR の重み，*s* はその PR の満足度である．PR の満足度とは，その PR の正の CE と適合する DE の重みの中の最小値であり，正の CE をもたない場合は1である．(ACTIVATE *a*) の実行前に *a* が WM に存在しない場合は w_0 は0とみなされる．*a* を新しく結論として WM に置かれる DE とすれば，*a* の重みは $s \times r$ として計算され，これを結論 *a* の確実性の値として用いることができる．これは，MYCIN における AND 節に対する確実性係数の計算式と同じ意味をもつものである．

PR, DE には，重みに加えて，付随的な情報として，3個のフラグ情報を付与することができる．PR には，もう一つ特別なフラグとして，PR が起動されたことがあるかどうかの情報を保持するためのフラグが用意されている．フラグは0または1の値をとる．

前章3.2節の競合解消のところでも述べたように，PR の重みの変更は，その PR の，モジュール内での位置を変えることになる．これは，状況に適応して PR の重みを変えながらモジュール内の構成を変えてゆくことによって，自己を再組織化してゆくプログラムの作成を可能とする．APSH のこのような機能を利用した，学習するプログラムとして，ハノイパズルプログラムなどを作成した．6章で簡単に紹介する．

5. プログラムの作成と編集

PR モジュールおよびデータの作成と編集は，基本的にはエディタを起動して編集用サブコマンドを用いることによって行う．PR モジュールおよびデータのデータ構造はリストなので，編集用サブコマンドはリストの編集向きに工夫されている．たとえば，EI (Element Insert), EC (Element Change), ED (Element Delete), ET (Element Transport), および EL (Element List) などのサブコマンドによる，リストの構成要素を単位とする編集などが可能である．

また，プログラムの構成要素である行為をコマンドとして用いることもでき，これによりプログラムの修正やデータの作成を行うことができる．このような APSH の機能は，従来の「エディタによるプログラム修正の可能なプログラミングシステム」に対して，「エディタによらないプログラム修正の可能なプログ

```
< ATTEND >
FACT 2;

PR   3 WAS CREATED.
     (3 (CUL 1) => (DEP (SOL 1)) (RETURN))
PR   4 WAS CREATED.
     (4 (CUL 2) => (DEP (SOL 2)) (RETURN))
KOTAE= 2
< ATTEND >
EDIT F;

< EDIT >
L

   1 (F)
   2 (4 (CUL 2) => (DEP (SOL 2)) (RETURN))
   3 (3 (CUL 1) => (DEP (SOL 1)) (RETURN))
   4 (1 (CUL 0) => (DEP (SOL 1)) (RETURN))
   5 (2 (CUL ?C) => (ADD ?C -1 ?A) (REP #1 ?C ?A) (F) (MULTSOL ?C)
       (MAKERULE ?C) (RETURN))
D 2 3

R

   1 (F)
   2 (1 (CUL 0) => (DEP (SOL 1)) (RETURN))
   3 (2 (CUL ?C) => (ADD ?C -1 ?A) (REP #1 ?C ?A) (F) (MULTSOL ?C)
       (MAKERULE ?C) (RETURN))
< ATTEND >
FACT 2;

PR   3 WAS CREATED.
     (3 (CUL 1) => (DEP (SOL 1)) (RETURN))
PR   4 WAS CREATED.
     (4 (CUL 2) => (DEP (SOL 2)) (RETURN))
KOTAE= 2
< ATTEND >
CHECK F (DEP (SOL 1))(RETURN);

< ATTEND >
REMRULE;

PR   3 WAS REMOVED.
< ATTEND >
CHECK F (DEP (SOL 2))(RETURN);

< ATTEND >
REMRULE;

PR   4 WAS REMOVED.
< ATTEND >
LIST F;

     (F)
     (1 (CUL 0) => (DEP (SOL 1)) (RETURN))
     (2 (CUL ?C) => (ADD ?C -1 ?A) (REP #1 ?C ?A) (F) (MULTSOL ?C)
       (MAKERULE ?C) (RETURN))
```

図 9 エディタと基本モジュールによる編集
Fig. 9 Editing by the editor and primitive modules.

ラミングシステム」としての一つの可能性を示したということができる．これは，PM，WM への動的アクセス可能な行為の柔軟さによって生じたものである．

図 9 は，P-FACT のモジュールFの編集をエディタのサブコマンドおよび基本モジュールによるコマンドを用いて行った例である．“EDIT F;” により編集状態〈EDIT〉に入り，サブコマンド“R” で編集結果を出力し〈ATTEND〉へ戻る．Lはリスト出力のためのサブコマンドである．“D 2 3” は削除のための編集用サブコマンドで，2番目から3番目までのリスト (PR-3 と PR-4) の削除を指示している．同じ処理を，CHECK により削除する PR を指定し，REMRULE によりその PR を削除することによって行えることも示している．

6. むすび

APSH で作成されたプログラムには，さらに次のようなものがある．

① 文字列予測プログラム：有限個のアルファベットの列を与えられて，後続する文字列を予測するための規則を発見するプログラムである[10]．16個のモジュールから構成され，PR の生成・付加と削除を反復しながら，正しく予測を行うための PR を発見する．

② ハノイパズルプログラム：3枚の円盤のハノイの塔問題を解決するためのよりよい手続きを学習するプログラムである．10 個のモジュールから構成され，問題を解くための初期知識を与えられ，それをもとにして，実際に問題解決の試行を行いながら，問題解決のための知識を獲得してゆき，しかも，試行回数が増すにつれて，問題解決のための知識をよりよい構成をもつように変えてゆく．知識の獲得は PR の生成・付加により行い，よりよい構成への変化は，状況に適応して PR の重みを変化させることにより行っている．よりよい構成とは，冗長な手順をとらずに，できるだけ短い手順で問題を解くことのできるような知識の構成である．

③ その他，文字列パターンを識別するための規則を帰納的に学習するプログラム，人間のメロディ認知プロセスモデルのプログラムなどの作成を行っている．

APSH によるいくつかのプログラム作成を通して，PM のモジュール構造化，基本的な適応・学習機能，および対話型プログラミングシステムとしての構成の有効性をほぼ確かめることができた．今後は，より多くのプログラミング経験を積むことに加えて，PR の一般化のための機能，PR のもつ文脈の処理，競合解消の規則，競合解消とモジュール呼出しにおける後戻り処理，そして PS におけるプログラミング支援ツールのあり方などについての考察を進めていきたい．なお，APSH は FORTRAN 77 で約 2,800 ステップのプログラムである．

謝辞 本研究を進めるにあたり，終始，貴重なご助言をいただいた本学工学研究科情報工学専攻情報システム工学講座の竹村伸一教授，APSH の機能決定に関してご討論いただいた本学文学部行動科学科認知情報学講座の阿部純一講師に感謝の意を表します．

参考文献

1) Newell, A.: Production Systems: Models of Control Structures, in Chase, W.G. (ed.): *Visual Information Processing*, pp. 463-526, Academic Press, New York (1973).
2) Shortliffe, E.: *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York (1976).
3) Rychener, M. D.: *Production Systems as a Programming Language for Artificial Intelligence Applications*, Department of Computer Science, Carnegie-Mellon University, Pittsburgh (1976).
4) 佐藤泰介：プロダクションシステムの試作とその使用経験，情報処理学会人工知能と対話技法研資，3-1 (1978).
5) Lenat, D. B. and McDermott, J.: Less than General Production System Architectures, Proc. IJCAI 79, pp. 928-932 (1979).
6) Davis, R. and King, J.: An Overview of Production Systems, in Elcock, E. W. and Michie, D. (eds.): *Machine Intelligence*, Vol. 8, pp. 300-332, Ellis Horwood Limited, Chichester (1977).
7) Rychener, M. D.: Control Requirements for the Design of Production System Architectures, Proc. Symp. AI and PL, pp. 37-44 (1977).
8) Goldstein, Ira P. and Grimson, E.: Annotated Production Systems: A Model for Skill Acquisition, Proc. IJCAI 77, pp. 311-316 (1977).
9) 溝口理一郎，大上勝也，富田雅也，西山静男，角所 収：階層的プロダクションシステム—HIPS，情報処理学会論文誌，Vol. 23, No. 2, pp. 177-186 (1982).
10) Waterman, D. A.: Adaptive Production Systems, Proc. IJCAI 75, pp. 296-303 (1975).
11) 斎藤正男，溝口文雄：知的情報処理の設計，コロナ社，東京 (1982).
12) 安西祐一郎，佐伯 胖，難波和明：LISP で学ぶ認知心理学 2，問題解決，東京大学出版会，東京 (1982).
13) Myers, G. J.: *Reliable Software through Composite Design*, Mason/Charter Publishers, Inc., New York (1975) (国友，久保共訳：高信頼性ソフトウェアー複合設計，近代科学社，東京 (1975)).
14) Winston, P. H.: *Artificial Intelligence*, Addison-Wesley, Reading (1977).

(昭和57年 9 月 8 日受付)
(昭和58年 5 月10日採録)