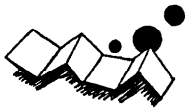
解　説



プロダクションシステムとその応用†

辻　井　潤　一††

1. はじめに

プロダクション・システム（略して PS）は，人工知能あるいは知識工学といった分野で近年盛んに使われ，大きな成果を収めてきているプログラム技法の1つである．特に，人工知能研究の成果を，現実的な場面に応用してゆこうとする知識工学の研究のほとんどが何らかの形で PS と関連を持っている．

知識工学については，本号の別項で詳細に解説されているので，ここでは触れないが，この分野における成果は，医療診断システム（MYCIN[24], MEDICO[26], PUFF[9]），有機化合物の構造推定システム（DENDRAL[8]），数学の新しい概念を作り出したといわれる AM[11]，分子生物学の実験をアレンジする MOLGEN[12]，タンパク質の分子構造を推定する CRYSALIS[7] 等，めざましいものがある．このような知識工学の分野では，各専門分野の研究者が持っている知識を，いかに簡単に計算機システムに移行できるかが重要となる．しかも，各専門分野の研究者が持っている知識は，医療診断システムの例を考えてみればわかるように，即座に計算機プログラムとして実現できるほど，知識相互間の関係が整理されていないのが普通である．このような分野では，まず知識を組み込むための枠組だけを用意し，専門家が徐々に新たな知識を入力することによって，システム全体の能力を向上させてゆくといったアプローチが望ましい．この枠組を提供するのが PS である．知識を計算機システムに組込んでゆく過程が広い意味でのプログラミングであると考えると，PS は知識工学の分野におけるプログラミングの手法を提供している．このような分野に対するプログラミング手法は，従来のプログラミング手法とは当然異なっている．

ある情報処理を行う場合に，従来のプログラミング手法では，まずその処理に含まれるコントロールの流れに注目する．この方向からのアプローチは，処理に本質的に含まれるコントロールの流れをそのままプログラム構造にひき移し，本質的ではなく単にプログラム作成の便宜のために導入される無意味なコントロールの移動（GO TO）は極力排けるべきだという考え方に発展し，構造化プログラミングの手法として定式化されてきている．しかしながらこの手法は，前述の知識工学の分野における情報処理のように，全体のコントロールの構造が不明確な分野，あるいはそれが事前にはとても決定できないような複雑な分野には，全く適用できない．PS は，このような分野に対するプログラミングの手法である．すなわち，コントロールの流れを前面に出してプログラムを考える方法に対して，PS はコントロールの流れを表面には出さずに，知識の単位に対応するルールの集合で処理の内容を記述してゆこうとする立場である．従来のプログラミング手法が Turing Machine (TM) のモデルで計算 (computation) を把えているとすれば，PS はルールの集合で計算を把える Post Machine (PM) のモデルに従ってこれを把えている．理論的に見れば，TM と PM は等価な計算能力を持っていることが証明されているが，工学として，あるいはプログラムを作る立場からみると，どちらの枠組に従うかは大きな相違をもたらすことになる．

前述のように，PS は知識工学の分野でめざましい成果を収めてきているが，一方では，音声認識システムにこれを応用して失敗したという例[14]も報告されている．PS 的なアプローチが成功する分野と，従来のプログラミング手法の方が適している分野とがある．現在，PS は様々な分野に適用されて，その特徴と限界が徐々に明らかになってきている段階である．いわば，PS は名人芸の段階から技術に近い段階になったといえる．

本解説では，まず PS の基本モデルについて述べ，次にこのモデルに従って PS が従来のプログラミング手法と比べて，どのような特徴を持っているか，また

† Production Systems and Their Applications by Jun-ichi TSUJII (Department of Electrical Engineering Kyoto University).

†† 京都大学工学部電気工学第二教室

PS が適した分野とそうでない分野等について考えることにする.

2. PS のモデル

2.1 基本モデル

PS は，次の3つの基本的な構成要素からできている[16].

(1) プロダクション・ルール（以下Pと略す）の集合 (PM—Production Memory)
(2) データ・ベース (WM—Working Memory)
(3) インタプリタ (PSI)

この3つの要素以外に，外界からの信号（刺激）を受け入れて WM に記号化して書き込む機構 Sensor と，PS からの指令にもとづいて外界に対して働きかけ（反応）を行う機構 Motor とがあり，全体の構成は図-1のようになる．Sensor および Motor は，外界とのインタフェースを行う部分であり，対象とするシステムによって，その構成は大きく変化する．(1)・(2)・(3)の PS の基本要素は，主として記号的な処理を行うための枠組であり，Sensor と Motor は記号的な処理以外の，記号化される以前の信号を記号化したり，記号的な処理から導かれたある種の結論を現実の行動に結びつける機構である．このようにすると，現実に取扱う対象が何であろうと，PS が対象とするものは完全に記号化されたものであると考えることができる.

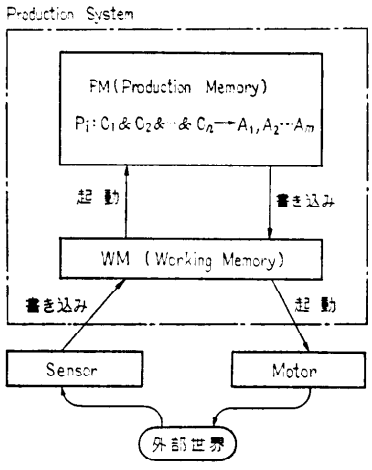


図-1 Production System の基本構成

このような前提のうえで，PS は記号列の変換を行う一般的なメカニズムを提供している.

PM に貯えられている P は，その P が起動される条件 (C_i) の集合と，条件が満された時に実行される動作 (A_i) の集合の対からできている．条件の集合部分をその P の LHS (Left Hand Side)，動作の集合を RHS (Right Hand Side) と呼ぶ.

PM 中の各 P は，常に WM 中の状況を監視しており，LHS が満足される状況が WM 中に生じれば，その時点で起動され RHS で指定された行動が順次実行されることになる．RHS で示される行動の系列は，一般に WM の状況を変化させることになるので，これによってひき起される WM の変化が，また別の P の LHS を満すことになり，次々と P が起動されてゆく.

ある WM の状況の下で，LHS が満足される P は一般には複数個存在することが考えられ，このうちどの P を起動するかを決定する必要がある．このようにある時点で，同時に起動可能な状態になっている P の集合を競合集合 (conflict set)，この中からどれか1つの P を選択することを競合の解消 (conflict resolution) と呼ぶ．この部分の作業は，すべて PS のためのインタプリタ (PSI) が行うことになる．現在，様々な PS の変種が開発されているが，この競合の解消に関してもいろいろなストラテジーが考案されている[13),22)].

以上のことから，PS の基本動作は次の3つのステップを繰返し実行することからできていることがわかる.

(1) PM 中のすべての P の LHS を，WM の状況と照合し，起動条件が満足されている P の集合（競合集合—conflict set）を作る.
(2) 競合集合の中から，ある種のストラテジーによって，起動すべき P を選択する（競合の解消—conflict resolution).
(3) 選択された P の RHS 部分を実行する.

(1)と(2)のステップを，現在の WM の状況で起動できる P をみつけ出すステップと考えれば，基本的には P の選択とその実行という2つのステップを繰返し実行することになる．この繰返しのことを recognize-act cycle と呼んでいる．PS は，この recognize-act cycle の繰返しで，仕事を進めてゆくことになる.

ここまでの PS の基本モデルは，一応現在 PS と称するほとんどのシステムに共通している枠組である．しかしながら，このモデルで実際のシステムを構成してゆくためには，この recognize-act cycle の各局面をもっと明確にしなければならない．PS は現在，適

用する対象分野や研究の目的等が異なるさまざまな立場から研究が行われており，どのような立場に立つかによって，設計方針が大きく異なっている．したがって，PSの具体的な構成例を見る前にまず次節ではPSに対する研究態度を整理しておく．

2.2 PS に対する基本的な立場

PS研究の流れの中には，PSを計算機構のモデルと把える立場[19]と，PSを人間の心理モデルを作るための枠組として見る立場[16]がある．さらに，この2つの立場は，理論を重視するかパフォーマンスを重視するかで，それぞれ2つの流れに分けられる．したがって，PS研究は4つのグループに大きく分けて考えることができる．

心理モデルを目指す研究は，理論かパフォーマンスかで，次の2つのグループにわかれる．

(1-A) 心理モデルの作成

(1-B) 知的な行動を示すシステムの作成（知識工学）

(1-A)が理論を，(1-B)がパフォーマンスを重視する立場である．(1-A)では，Newell等の研究に典型的にみられるように，人間の記憶・認知のモデルを，実験心理学の成果を基礎にして，計算機的に実現しようとする．これに対して，(1-B)はいわゆる「知識工学研究」の立場[9]であり，対象分野を限定し，その分野で専門家と同じ振舞いをするシステムを作ることを主目的としている．この結果，(1-A)と(1-B)では，PS中に入れられるルールPを，どのような単位とするかで，大きく異なってくる．(1-A)では，人間の認知機構という目には見えない未知のメカニズムを対象としているため，各Pは人間の無意識下の操作に対応させられる．一方，(1-B)では専門家が，その分野でひとまとまりと考える知識を，自然な形でPSに移すことを目的としているので，専門家が意識的に行う操作(推論)の基本的な単位をPとして選ぶことになる．

専門家が人間である以上，その知的活動も人間のもつ一般的な認知機構に支えられているはずであり，(1-B)で1つのPとして表現されるものも，より詳細にみれば，(1-A)のPの集合によって説明することが可能であろう．しかし，知識工学の立場からは，これをより細いPに分解して考えることはしない．

一般に，(1-A)のPSでは，Pとして選ぶ単位が細く，WMに対する操作も単純な操作に限定されるのに対して，(1-B)ではその逆の傾向がある．また，(1-B)では，分野固有の多くの操作が，PSの裏側で行われることになり，対象とする分野が異なれば，ちがったPSが作られることになる（知識工学で作られているシステムの多くは，PSの基本的な発想だけを借用し，その中に分野固有の様々なヒューリスティクスを埋め込んだPS的システムになっている）．

(1-A)，(1-B)が，「人間（あるいは専門家）の知的活動」のモデル化を目指すのに対して，PSを計算機構としてみる立場がある．

(2-A) 計算機構の理論としてのPS

(2-B) プログラミング言語としてのPS

(2-A)の典型は，POSTによって示されたPost Machineで，この枠組の計算能力はTuring Machineと同じであることがわかっている[6]．この研究は，現在の各種のPSの理論的基礎を与えている．Turing Machineの基本動作をそのままプログラミング言語として使うことができないと同じように，このPost Machineを基礎にして，いろいろな機能を入れることによって，汎用のプログラミング言語が開発されている（OPS[13]，POPS[29]，PSnlst[21]）．Rychenerは，PSnsltを使って，これまでの人工知能分野の典型的なシステム（STUDENT，GPS，MILISY，EPAM等）の書換えを行っており，PSが人工知能用のプログラム言語として有効であることを示している．

以上4つの研究の立場によって，構成されるシステムは様々な変化をうけることになるが，本解説では一応理論的に完成している(2-A)の立場については触れず，(1-A)・(1-B)・(2-B)について考えることにする．また，(1-A)についての記述も参考程度にとどめ，主として後の2つの立場について述べる．

3. PS の記述例

PSについての具体的なイメージを得るために，各立場からの典型的な記述例をみてみることにしよう．**図-2**はNewellのPSGの例である．この例でNewellはSteinbergの心理実験*を説明するモデルを提案している．図中PD1からPD4が，このためのPである．被験者へ提示された数字の集合が(ELM 1)(ELM 4)(ELM 9)，テスト用数字が(PROBE 9)でそれぞれWMに与えられる．テスト用数字が集合中

* 被験者にまず数字の集合を提示し，次に1つの数字を示して，その数字が初めに与えた数字の集合に入っているかどうかについて答えさせる．この時の反応時間，人間の犯す誤り等をみることによって，人間の短期記憶（Short Term Memory—STM）に関する様々な仮説が立てられている．NewellはPSのWMを，このSTMのモデルと考えて，計算機モデルを作成している．

```
00100 〈DIGIT〉: (CLASS 0 1 2 3 4 5 6 7 8 9)
00200 ANY: (VAR)
00300 ;
00400 RESPOND: (ACTION (NTC (RESPONSE ANY))
      (SAY ANY) (OLD**))
00500 ATTEND: (OPR CALLTOUSER)
00600 ;
00700 PS ST1: (PD1 PD2 PD3 PD4)
00800 ;
00900 PD1: ((PROBE) AND (OLD(RESPONSE))→(OLD**))
01000 PD2: (PROBE 〈DIGIT〉) AND (ELM 〈DIGIT〉)
          →(RESPONSE YES)
01100     RESPOND)
01200 PD3 : ((PROBE) AND (ELM)
          →(RESPONSE NO) RESPOND)
01300 PD4: (READY→ATTEND)
01400 ;
01500 STM: (READY (ELM 1) (ELM 4) (ELM 9) NIL NIL)
01600 ;
```

(注) PD1 から PD4 が，P. 2つの Motor に対する操作 RESPOND と ATTEND が行番号 400 と 500 とで定義されている．

PD4 によって，人間との応答が起動され，調査用数字が (PROBE 〈DIGIT〉) の形で与えられる．これは，WM (図では STM と記されている—行番号 1500) の左端に記入される．

PD2 が YES, PD3 が NO の答えを返すためのPで，PM 中の P に優先度があるために，PD2 が満足されない時にはじめて PD3 が起動される．

PD1 は，終了のための条件，行番号 400 の ACTION によって，YES または NO の返答をした後に STM 中に (OLD RESPONSE) の表現が記入される (**は，STM の先頭の要素を指す) ので，これによって，PD1 が起動される．

図-2 PSG による記述例 (Steinberg の心理実験)

のメンバであるかどうかは，PD2 で判定される．この P 中の 〈DIGIT〉 が変数の役割を果している．PSG の特徴を整理すると，

(1) WM のサイズに制限がある (図では6)．

(2) PM 中の P は優先度順に並んでおり，最初に WM の内容と照合のとれたPが起動される．

(3) WM の要素にも，左から右への優先度がある．WM 中の複数の要素が，あるPの LHS と対応がとれるときは，より左の要素が優先される．また，照合のとれた要素は WM の最左端に移動される (再想起の機能)．

(4) RHS に書かれる記述は，Motor に対する記述 (図の例では ATTEND と RESPONSE) 以外は，すべてそのまま WM に書き込まれる．書き込みは WM の最左端に対して行われる．

(5) WM の要素は，サイズ以上になると右端の要素から順に消されてゆく．

Steinberg の実験に使われた仕事自体は単純で，通常のプログラム言語でも数行で実現できる．Newell は，この仕事を実現するプログラムを作ることを目的としたのではなく，人間の記憶モデルの作成を目指しており，この単純な PS はさらに検討が加えられ，人間の反応時間との対応が論じられることになる．例からもわかるように，PSG は，LHS, RHS の記述において非常に単純な枠組をとっているのが特徴である．この特徴は，Newell が音声理解システム HARPY を対象にして構成した PS (HPSA 77)[17] にもひきつがれている．

```
PREMISE  ($AND(SAME CNTXT INFECT PRIMARY-
                BACTEREMIA)
               (MEMBF CNTXT SITE STERILESITES)
               (SAME CNTXT PORTAL GI))
ACTION   (CONCLUDE CNTXT IDENT BACTEROIDES
               TALLY 7)
```

If (1) the infection is primary bacteremia, and
(2) the site of the culture is one of the sterilesites, and
(3) the suspected portal of entry of the organism is the gastrointestinal tract,

then there is suggestive evidence (.7) that the identity of the organism is bacteroides.

(注) ACTION 中の .7 の数字は，この診断の CF (Certainty Factor) で，この数値と PREMISS の各評価値とから，診断全体の確信度が計算される．PREMISS 中で，$AND でくくられる各条件表現は，図-4 の定形的なフォーマットに従って記述される．

図-3 MYCIN の診断ルールの例

次に，(1-B) の代表的な PS として医療診断システム MYCIN の例を図-3 に示す．MYCIN は，約 200 のこのような P からできている．図中 PREMIS が LHS, ACTION が RHS である．PSG と対照的に，大きな単位の知識が1つの P で表わされている．PS の立場から見て，MYCIN の特徴を整理すると次のようになる．

(1) WM と LHS の照合が，単純なパターン照合 (pattern matching) ではなく，PREMISS 中の各表現に対応した専用の LISP 関数によって実現されている．

(2) 各関数は，患者についての各種検査データやそれまでに下された診断結果を参照して，各 PREMISS がどの程度に満足されたかを −1 から +1 の間の数値で返してくる．PREMISS 中の各項が $AND あるいは $OR でくくられている場合は，各関数の返した評価値の最小値あるいは最大値が PREMISS 全体の評価値とされ，P につけられた CF (Certainty Factor——図の例のPでは 0.7) との積がとられ，結論の確からしさが計算される．一種の近似推論を行う．

(3) WM には，患者についての検査情報およびそれまでの推論から得られた中間的な診断結果が記憶されており，サイズの制限はない．

(4) システム中の P は，すべて図-4 に示すような定形的な形式に従って記述されている．

⟨predicate function⟩ ⟨object⟩ ⟨attribute⟩ ⟨value⟩

(注) MYCINでは,条件表現を上記の4字組で表わす. ⟨predicate function⟩として 24, ⟨object⟩として 11, ⟨attribute⟩として 80 のものが標準として用意されている.

図-4 MYCIN における条件表現のフォーマット

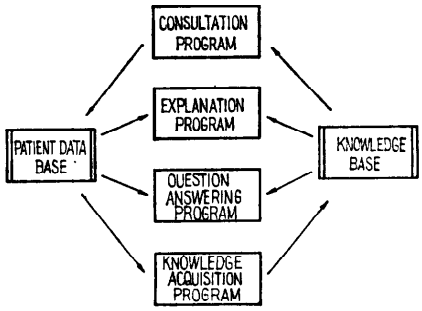


図-5 MYCIN のシステム構成図

(1),(2),(3)の特徴は, MYCIN が PSG のような PS の純粋な枠組だけでなく, 医療診断という分野固有の性質をできるだけ反映し易いように様々な機構を組込んでいることを示している. 特に, PREMISS と WM の照合に分野固有の操作を LISP 関数の形で埋め込んでいることが注目される.

MYCIN の PS 的な性質は(4)の特徴にあらわれている. **図-5** に MYCIN の全体的な構成図を示すが, この図からもわかるように, MYCIN では単に診断を行うだけでなく, それ以外に, EXPLANATION プログラム (診断の過程を人間に説明する) や KNOWLEDGE-ACQUSITION プログラム (新たな診断用知識の入力を行う) が, 重要な役割をうけもっている. 診断プログラム以外のこのようなプログラムは, PM 中のPをあたかも通常のデータであるかのようにみて, それを参照したり, 操作したりする. PM 中のPは, 診断過程では一種のプログラムとして働くと同時に, 他のプログラムによって参照・操作されるデータとしての性質も持っていることになる. このようにPがデータとしての性質を持つためには, Pがある一定の定まった形式に従って記述されていることが必要となる.

分野固有の操作を含む例として, もう1つ(1-B)からの例を**図-6** に示す. この例は, DENDRAL で使われたPである. LHS は, 単なる文字列 ESTROGEN であるが, 実際には ESTROGEN と名付けられた分子構造 (ネットワーク構造で記憶されている) を指示しており, RHS で指定された各操作もこのネットワーク構造中のリンクやノードを参照する形で, 記述されている.

```
ESTROGEN ⇒ (BREAK (14 15) (13 17))
            (HTRANS +1 +2)
```

図-6 DENDRAL の P の記述例

MYCIN, DENDRAL では, P 中の各表現に対して, それをサポートする分野固有の操作が組み込まれていた. また, 競合の解消等のコントロールに関しても, それぞれの分野に応じた機構が固定的にえらばれている. この傾向は(1-B)すなわち知識工学における研究に共通してみられる特徴である. これに対して, これらを一般的に把え, Pの記述を変えることによって, コントロール構造を含めて様々なシステム構成がとれるようにしようとする試みがある. (2-B)の汎用プログラミング言語の開発を目指す立場である.

```
I1; "INIT 1" :: INIT(P)
  ⇒EXISTS (MNK, BAN, BX 1, BX 2, BX 3)
    & LOC (MNK, 1,1,1) & LOC (BAN, 5,5,3)
    & LOC (BX 1,7,8,2) & LOC (BX 2,7,8,1)
    & UPON (BX 1, BX 2) & LOC (BX 3,4,6,1)
    & ISMONKEY (MNK) & ISBANANAS (BAN)
    & ISBOX (BX 1) & ISBOX (BX 2) & ISBOX (BX 3)
    & HVAL (BX 1,3) & HVAL (BX 2,4) & HVAL (BX 3,5);
G1; "GOTO CK" :: GOTO (M,X,X) & LOC (M,X 2,Y 2,H)
    & SATISFIES (H,H EQ 1)
  ⇒LOC (M,X,Y,H) & NEGATE (ALL);
G2; "GOTO CLIMB" :: GOTO (M,X,Y) & LOC(M,X2,Y2,H)
    & SATISFIES (H,H?.GREAT 1)
  ⇒CLIMBDOWN (M) & GOTO (M,X,Y);
R2; "REACH-" :: REACHFOR (M,B) & LOC (M,X,Y,H)
    & LOC(B,X,Y,H2) & SATISFIES 2(H,H2,H?.LESS H2)
    & NOT EXISTS (HN) & CLIMBUP (M,X,Y,HN))
  ⇒NEEDBOX (M,X,Y,H) & NEGATE (1);
```

(注) I1は初期設定のためのP.起動されると,LISP関数EXISTSが実行され, 変数 MNK, BAN, BX 1, BX 2, BX 3 に対応する定数 (instance) が作成され (例えば, MNK に対して MNK-1), ISMONKEY (MNK) 等のそれ以外の述語表現に対応するインスタンス表現 (ISMONKEY (MNK-1)) がWMに設定される. EXISTS は, 単なるリスト表現を WM に書き込む通常の action とはタイプがちがう.

G1, G2 は, monkey の場所移動についての P, G1 は, GOTO action の前提条件が満足される場合で, 即座に action が実行されるのに対して, G2 は GOTO というゴールを, CLIMBDOWN というサブゴールに分解している. LHS 中の SATISFIES は, R2 中の SATISFIES 2 と同様に, 評価可能な述語表現を入れるための特別な表現.

図-7 PSnlst による P の記述例 (monkey and banana のための P)

図-7 は, Rychner の PSnlst で AI の古典的問題 monkey and banana を解くための P の記述例である. PSnlst の特徴を列挙すると,

(1) LHS には, GOTO・LOC のような通常の述語 (通常のパターン照合で WM との照合がとられる) の他に, SATISFIES と呼ばれる特別な述語が用意されている. 図中 SATISFIES (H H EQ 1) は, 述語 EQ に対して定義されたプログラムを起動し, (H EQ

1) の真・偽を検査することを指示している．(H EQ 1) は，WM 中の直接の要素としては存在していない．EQ の位置には，任意に定義した LISP 関数が使用できる．

（2） WM と LHS の照合は，SATISFIES のような特殊な表現を除くと，パターン照合によって行われるが，Newell が PSG に対してとった立場と異なり，AND, OR, NOT を含む複雑な論理表現を使って，しかも任意個の変数を入れることができる．

（3） WM 中には，(GOTO MNK X Y) のようにゴールに関する情報，および (LOL MNK X Y H) のように問題解決過程における状態記述に関する情報，の双方が置かれている．GPS や STRIPS のような通常の問題解決プログラムでは区別されるこのような2種の情報を共通の WM におくことにより，前向き推論・後向き推論が同一の機構で実現される．

（4） PSG では，WM からの要素の消去がシステムによって自動的に（WM—短期記憶—のサイズの制限によって）行われたが，PSnlst ではこれをプログラマが明示的に指定する．図中，RHS に使われた述語 NEGATE がこの役割を果す．NEGATE (数字) は，LHS の数字番目の表現と照合のとれた要素を WM から消去する．NEGATE (ALL) は，LHS の表現と照合のとれたすべてを WM から消去する．

この他，PSnlst を実用的な AI 用言語とするために，WM に変化が生じた場合，その変化によって起動条件 (LHS) が満足される可能性がある P をすべて取り出してスタックする機能，特殊な制御機構 (iteration, P の sequencing) を実現するための機構等も用意されている．日本においても，電総研の佐藤が PS を AI 用言語として採用する立場から，PS と述語論理あるいは PROLOG 等との関連を論じている[29]

4. PS の特徴

本章では PS をプログラミング手法とみた場合に，通常のプログラミング手法と比較して，その特徴を整理する．PS の処理は，前述のように recognize-act cycle の繰返しである．WM には，それまでに行われた処理のすべての結果が反映されており，recognize のステップで，これを参照することにより，次に起動すべき P が決められる．この段階では，すべての P が起動され得る資格を持っている．

これに対して，通常のプログラムでの処理は，ほとんどの場合，あらかじめ決められた順序で action を実行することによって進められる．action と次の action の間に recognition のステップは必要ではない．プログラムの流れの中で，次に実行する action はほとんど常に一意的に決まっている．通常のプログラムにおいても，条件判定を行い，次の action の選択を行うこともあるが，この場合でも次にとり得る action の集合は極めて限定されている．すなわち，プログラム中のどこにコントロールがあるかによって，どこまで処理が進んだかが明確に把握されており，次の action がコーディングの時点でプログラマによってあらかじめ決定されている．

これに対して，PS ではコントロールの流れが事前には決められていないので，それまでの処理の全過程を反映している WM と，その WM の状況から判断して次の action を決定する recognition のステップが必要となる．

このことから，PS 向きの分野とそうでない分野とを次のように整理できる．

（1） コントロールの流れが簡単に把握でき，次の action が事前に決まる分野には不適当である．この場合には，通常の手法の方がはるかに効率が良い．

（2） 複雑なコントロール構造が本質的な分野には不適当である．コントロールの流れをプログラマが意識的に把握していてはじめて可能な recursion や iteration が重要な役割を果す分野，あるいは process と subprocess のように処理単位の間に深い相関があり，立体的なコントロールを必要とする分野には不適当である．

（3） 逆に，分岐が頻繁におこなわれてコントロールがプログラム中のどこにあるかが次にとるべき action の決定にほとんど役に立たず，次の action を決めるためには，それまでの処理の状況全体をみることが必要であるような分野には有効である．

もちろん，コントロールに関する情報をも WM に入れれば，（1）・（2）の分野にも適用できる[21]．また PS の考え方に，ペトリネットの手法を入れようとする試み[27]もあるが，むしろ PS の本来の特徴を損う危険があろう．PSnlst の例で，状態記述の他にゴールに関する記述を WM に入れているが，これは PS の枠組の中で自然にコントロールに関する記述を入れる手法だと考えても良い．

このように，PS は従来のプログラミング手法ではとても事前にコントロールの構造が決定できない「複雑な」対象に適しているが，この「複雑さ」が高度に

いり込んだコントロール構造を要求するという意味での「複雑さ」であってはならない．従来の意味でのコントロールを入れても，CASE 文や IF 文の連続にしかならないような分野のプログラミングに適している．しかも PS が最も有効に働くためには，recognize における分岐が意味を持つ程度に十分に多様性のある action 系列が存在しなければならない．すなわち，PM に貯えられるPの数が十分大きく，各Pがほぼ対等で，あるPが起動された後には必ず一連のPが起動されるといったP間に強い相関がない時に特に有効となる．

Davis[2]，佐藤等[29]は，PS の利点として，特に次のような項目をあげている．

（1） Modurality：PM 中の各Pは，WM との関係だけを考えて記述できる．P間の相互干渉は，WM への書き込みと参照を通じて間接的に行われるので，他にどのようなPがあるかを意識する必要がない．

（2） Readability：通常のプログラムでは，使われる知識が処理の流れの中に埋もれてしまい，全体のプログラムを解読しなければ，どのような知識がどのように使われているかを理解することが一般にできない．これに対して，PM 中の各Pは，それ自体が1つの完結した知識の単位に対応しているので，その意味がそのPを見るだけで理解できる．

（3） Self-Explanatory：処理の過程が，P の起動の連鎖となるので，なぜそのような結論に到達したかが，この連鎖をたどることによって説明できる．通常のプログラムでは，内部は完全なブラックボックスとなるので，このような能力を得ることは困難である．

上の(1)・(2)・(3)の利点は，知識工学で対象とするような，プログラム化すべき知識の全体像が事前に決定できず，発展的 (evolutional) に作成せざるを得ないシステムの場合には望ましい特徴である．

しかしながら，このような利点は本章のはじめに述べたように，PS に適した分野に適用した場合，しかも知識の適切な単位をPとして選択した場合に初めて得られるものであることに注意する必要がある．Newell の PSG の例からもわかるように，PS をプログラミング手法としてだけみれば，Steinberg の問題を解くのに PS の枠組を採用するのは馬鹿げたことである．また，DENDRAL や MYCIN が成功したのは，対象とした分野が，入力データをある手順に従って加工するという従来の計算機処理とはちがって，入力データの意味を解釈する（分子構造の判定，医療診断）ことを基本的な問題としていること，専門家が直観的にひとまとまりであると判断する知識の単位をうまくPSのPとして設定したことによる．DENDRAL でもし分子構造の結合・分離の操作自体も複数個のPに分けて表現したとしたら，上のような利点は得られなかったであろう．(2)・(3)の性質も，結局は単位として選んだPのレベルでのみ保障された性質である．また，この2つの性質を得るためには，MYCIN の項で述べたようにPをなるべく定形的な形式で書く必要があることに注意しなければならない．

5. 問題点と今後の方向

PS の問題点としてまず初めに考えられるのは，recognition ステップでのオーバヘッドであろう．通常のプログラミングでは，まったく必要のないこのステップが，action と action の間に必ず介在することになる．

初期の PS では，recognition はストリング列あるいはリスト表現同士 (LHS と WM) の照合で実現されていたのに対して，より広い現実的な問題を取扱う最近のシステムでは，評価可能な (LISP 関数で表わされた) 述語表現が LHS の記述として導入されている．パターン照合は，それ自体処理時間を要する操作であるが，すべての recognition がこの手法で行われるとすれば，LHS の各パターンを discrimination-net やパターンの hashing 等を工夫することにより，PM 中のすべてのP の LHS を対象とせずに，照合のとれる可能性のある少数のPだけを選択的に取り出すことも可能である．しかし，評価可能な述語表現を許すことになると，このような一様な記憶構造はとれなくなり，その都度各P の LHS を評価することが必要となって，recognition のオーバヘッドが増えることになる．

C. Rieger[20] は，この欠点を解消するために，LHS の記述をパターン照合による部分とそうでない部分に区別し，パターン照合による部分だけを triggering tree と呼ぶ一種の discrimination net に構造化することを提案している．

いずれにしても，LHS に実行可能な LISP 表現を許すことは，Pの記述の中に Sensor についての記述を入れることを意味しており，1章で述べた仮定「PS は完全に記号化された世界を扱う」に反することになる．Sensor に関する記述は，そのシステムが対象とする外部世界と直接関係しており，一般化が困難であ

る．知識工学における各システムが，PS のアーキテクチャを採用しながら，なおかつ一般的なシステムとはならず，個別システムの作成に終始している原因の1つは，この Sensor に関する記述が一般化できないことにある．したがって，PS がより一般的な知識記述の体系となるためには，PSnlst のように使用者定義の LISP 関数の使用を許すだけでは不十分で，Davis[5] が試みているように，P が取扱うデータに関する記述までも含めたより広い枠組を考えてゆく必要があろう．このことは言い換えると，PS は基本的には計算機構の枠組であって，あくまで手続的知識の1つの表現形であり，これがより一般の枠組となるためには，Pの取扱うデータに関する記述についても何らかの一般的枠組を用意する必要があるということになる．この方向は，PS 以外の人工知能研究，たとえば，意味ネットワークや Frame の研究と PS との接点を求めることにもなろう．

知的な情報処理が，PS の提供する Data-driven，あるいは Event-driven な側面と，通常のプログラミングのような control-driven な側面の2つを持っているとすれば，PS に何らかの意味でコントロールの概念を入れることが重要である．現在，PS は競合の解消[13]という形で備えつけのコントロール機構を持っているが，より大きなシステムを構成するためには当然不十分である．我々は，1つのPで表現される知識の単位よりもさらに大きな知識のまとまりを容易に相像できるし，各分野の専門家はそのようなまとまりがどのような問題に対して，いつ全体として役に立つかを知っていて，問題解決過程をうまく制御しているものと考えられる．このような，Pよりもさらに上位の知識のまとまりに関する知識，一種のコントロールに関する知識をも PS 的な形式で表現することが考えられる．Davis[4] は，このような発想から，Meta-Rule の考え方を提案している．この Meta-Rule の考えは，これまでプログラムの中に埋もれてしまっていた発見的知識（ヒューリスティックス）を，一段上位の別のクラスの PS として把えるものであり，今後の PS の方向として興味がある．図-8 に Meta-Rule の例を示す．

この他，本解説では触れなかったが，PS の各Pがプログラムの一部でありながら，他のプログラムによって操作されるというデータ的な性質を持っていることに注目して，Pを自動的に修正・増強する学習の問題[18]，Sensor や Motor とのより緊密なインタフェースを必要とする画像理解[15]やロボットの問題解決[30]に PS を適用する問題等が，今後の興味のある問題として残されている．

```
METARULE 002
If  1) the age of the client is greater than 60,
    2) there are rules which mention in their premise blue-chip
       risk,
    3) there are rules which mention in their premise specula-
       tive risk,
then it is very likely (.8) that the former should be used before
    the latter.
PREMISE
($AND (GREATER OBJCT AGE 60)
  (THEREARE OLRULES ($AND
    (MENTIONS FREEVAR PREMISE BLUE-CHIP)) SET 1)
  (THEREARE OLRULES ($AND
    (MENTIONS FREEEVAR PREMISE SPECULATIVE))
      SET 2))
ACTION
  (CONCLUDE SET 1 DOBEFORE SET 2 .8)
```

図-8 Meta-Rule の記述例

6. おわりに

コントロール構造を中心にした従来のプログラミングに対して，PS はまったく別の方向のプログラミング手法を提供している．Rieger[20] は，これを demand computation と spontaneous computation の区別として把えた．この2つの流れは，必ずしも人工知能研究という限定された研究分野だけでなく，Backus の Functional Programming，データ・フロー計算機の提唱にみられるように，従来の計算機構の把え方と，それを乗り越えようとする流れの対立というように，もっと広い視野でながめることもできよう．また，日本でも MECS-AI のような医療診断システム，日常生活の推論システム[39]等の PS の応用システムも積極的に作られるようになってきた．日本での今後の研究が期待される．

最後に，関係論文の収集において，電子技術総合研究所佐藤泰介氏にお世話になったことを記し，感謝する．

参考文献

1) Buchanan, B. and Michell, T.: Model-Directed Learning of Production Rules, in Pattern-Directad Inference Systems (eds. D. A. Waterman and F. Hayes-Roth), Acadimic Press, (1978).
2) Davis, R. and King, J.: An Overview of Production Systems, Stanford A. I. Lab. Memo, AIM-271 (1975).
3) Davis, R., Buchanan, B. and Shortliffe, E.: Production Rules as a Representation for

Knowledge-Based Consultation Program, Artificial Intelligence, Vol. 8 (1977).
4) Davis, R.: Generalized Procedure Calling and Content-Directed Invocation, in Proc. of Sym. on Artificial Intelligence and Programming Languages, SIGART Newsletter No. 64 (1977).
5) Davis, R.: Knowledge Acquisition in Rule-Based Systems-Knowledge about Representation as a Basis for System Construction and Maintenance, in Pattern-Directed Inference System (eds. Waterman and Hayes-Roth), Academic Press, (1978).
6) Denning, P. J., Denm's, J. B. and Qualitz, J. E.: Machines, Languages, and Computation, Chapter 14, Prentice Hall (1978).
7) Engelmore, R. S. and Nii, H. P.: A Knowledge-Based System for the Interpretation of Protain X-ray crystallographic data, Stanford Heuristic Programming Project Report, HPP-77-2 (1977).
8) Feigenbaum, E. A., Buchanan B. G. and Lederberg J.: On Generality and Problem Solving: A Case Study Using DENDRAL Program, Machine Intelligence 6, Edirburg Univ. Press (1971).
9) Feigenbaum E. A.: The Art of Artificial Intelligence: I. Themes and Case Studies of Knowledge Engineering, Proc. of 5th IJCAI, Cambridge (1977).
10) Hayes-Roth, F, and V. R. Lesser,: Focus of Attention in the HEARSAY-II Speech Uuderstanding System, Proc. 5th IJCAI, Cambridge (1977).
11) Lenat, D. : The Ubiquity of Discovery: 1977 Computers and Thought Lecture, Proc. 5th IJCAI. Cambridge (1977).
12) Martin, N. and Frieland, P. et, al.,: Knowledge Base Management for Experiment Planning in Molecular Genetics, Proc. 5th IJCAI, Cambridge (1977).
13) McDermott, J. and Forgy, C.: Production System Conflict Resolution Strategies, in Pattern-Directed Inference Systems (eds. Waterman and Hayes-Roth), Acadimic Press (1978).
14) Mostow, D. J. and Hayes-Roth, F.: A Production System for Speech Understanding System, in Pattern-Directed Inference Systems (eds. Waterman and Hayes-Roth), Academic Press (1978).
15) Nagao, M. and Tsujii, J.: S-net: A Foundation for Knowledge Representation Languages, Proc. 6th IJCAI (1979).
16) Newell, A.: Production Systems: Models of Control Structures, in Visual Information Processing (ed. W. Chase), Academic Press (1973).
17) Newell, A.: Harpy, Production Systems and Human Cognition, Department of Computer Science, CMU (1977).
18) Newell, A.: An Instructable Production System, in 'Pattern Directed Inference Systems' Waterman and Hays (Eds), Academic Press (1978).
19) Post, E. L.: Formal Reduction of the General Combinatorial Decision Problem, Amer, J. Math., 65 (1943).
20) Rieger, C.: Spontaneous Computation and Its Role in AI Modeling, in Pattern-Directed Inference Systems (eds. Waterman and Hayes-Roth), Academic Press. (1978).
21) Rychener, M.: Production Systems as a Programming Language for Artificial Intelligence Applications, ph. D Thesis CMU (1976).
22) Rychener, M.: Control Requirements for the Design of Production System Architectures, Proc. of Symposium on Artificial Intelligence and Programming Languages, SIGART Newsletter No. 64 (1977).
23) Rychener, M. and Newell, A.: An Instructable Production System: Basic Design Issues, in Pattern-Directed Inference Systems (eds. Waterman and Hayes-Roth), Academic Press (1978).
24) Shortliffe, E.: Computer-Based Medical Consultations: MYCIN, American Elsevier, New York (1976).
25) Vere, S. A.: Relational Production Systems, Artificial Intelligence, Vol. 8 (1977).
26) Walser, R. L. and McCormic B. H.: A System for Priming a Clinical Knowledge, Proc. N. C. C., (1977).
27) Zisman, M. D.: Use of Production Systems for Modeling Asynchronous, Concurrent Process, in Pattern-Directed Inference System (eds. Waterman 8 Hayes-Roth), Academic Press (1978).
28) 青木 泰太, 田中 幸吉,「相互作用を持つグラフプロダクションシステム」, 信学会論文誌 D, Vol. 61-D, No. 12 (1978).
29) 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).
30) 田中 穂積,「日本語の意味構造を抽出するシステムEXPLUSについて」信学会論文誌, Vol. 61-D, No. 8 (1978).
31) 溝口 文雄, 中村 公夫,「小規模プロダクションシステムについて」, 情報処理学会第19回全大 (1978).

(昭和54年5月16日受付)