

プロダクション・システムの考察

東京電機大学理工学部

上野晴樹

1 はじめに

プロダクション・システム (Production System, 略してPS) は、知識型 (Knowledge-based) システムあるいは知識表現言語の中で、最も歴史が古く、簡単かつ基本的なシステムである。我々はこれまでに2つの知識表現言語COMEX (Compact Knowledge Based Expert System) およびFMS (Frame Manipulation System) を開発し (9, 10, 14)、実用指向の研究と研究指向の研究を併行してきたが、今回あらためて2つのPSを開発しつつあるので、これからこの種のシステムの開発を計画されている方々の参考として、開発の立場から見たPSについて述べてみたい。

2 開発の目的

現在開発を進めているPSはSMARTおよびTRYである。いずれも汎用ツールとして開発しているが、SMARTは本格的な汎用ルール型知識表現言語をめざしており、FACOM M-160システムを用いてUTILISP (11) で記述している。

一方、TRYはマイコンPC-9800を用いてBASICで記述している。こちらの方は、エキスパート・システムの教育・普及を主たる目的としているが、比較的小規模な問題については、ある程度の実用性を持たせられるものと考えている。(名前の由来はTry yourselfである)。

TRYの記述にBASICを選んだ主な理由は、1) 現在マイコン用に普及しているBASICは基本的な文字処理機能を持っており、2) メモリ空間が100kバイト以上であり (cp/m の下でのFORTRANやLISPは64kバイトの空間であり、しかもcp/mと言語処理プログラムを入手しなければならない)、3) 漢字かなまじり文が使用できる。BASICの実行速度の遅さは妥協することにする。

「なぜ今更PSか?」という意見もあるが、IF-THEN形式で統一された知識表現は、たいていのプログラム言語がもっている基本的な判断文であり、しかも直観的に解り易いので、エキスパート・システムの入門者用として最適であるばかりでなく、小規模な意思決定プログラムを書くにはほぼ十分であると考えられる。これまでに開発された知識型エキスパート・システムをみても、DENDRALをはじめ、MYCIN、PUFF、VM、EXPERT、R1などなど、PSが最も多い。PSの弱点もすでに色々明らかとなっているので、適切に計画すれば存在理由はある。

3 歴史的考察

人工知能の研究者達が、推論中心から知識中心の研究へと方向転換を始めるきっかけとなったDENDRALはPSであり、70年代へ入って広く注目を集めたMYCINもPSである。その後汎用知識表現言語が続々ではじめるようになったが、PSが最も多い。すなわち、EMYCIN (13)、EXPERT、OPS、ROSIE、RITA、MECS-AIなどである。これらの言語が提供されたことによって、ルール型のエキスパート・システムの開発が容易となり、PUFF、R1、RHEUMなど様々のシステムが試作され、実際に使われ始めている。

これらを歴史的にながめると面白い事実が解る。PSはもともとAIの分野で開発されたアイデアではなく、コン

ピュータが現われるより前の1943年にPost (15)によって組み合わせ問題のformal reductionの理論として提案されている。いわゆる古典的PSについてはMinskyの著書(16)にゆずりここではふれない。さて、DENDRALは化学における分子構造を推定するためのエキスパート・システムであり、初期のPSとして極めて有名である。1965年にProjectがスタートして以来、現在でもまだ続いており、NMR(核磁気共鳴)へも応用されるようになった。このDENDRALはPSとして知られているが、最初からPSを意識していたのではなく、試行錯誤の結果PSの形をとるようになった様である(2)。それは、知識の表現をIF-THEN形式で統一することにより、理解し易く、定義・変更・拡張が容易であり、しかも推論メカニズムもスッキリした構造となるなど、柔軟性に富んだシステムが実現できるからであった。これは、システムの能力(performance)を高めることに焦点をしばって開発された結果であった。感染症に対する診断および抗生物質による治療計画を支援するためのMYCINは、同じスタンフォード大学のHPPで開発されているが、これは最初からPSとして設計された。すなわちDENDRALの経験が生かされている。

MYCIN(2)における試はDENDRALに比べてはるかに高度なものと言える。すなわち、DENDRALは能力のみに重点が置かれていたが、MYCINでは意思決定の支援に重点が置かれている。例えば、エキスパートからの知識の獲得を容易とするために自然言語的表現でルールを定義したり確認したりすることができることや、ユーザがMYCINを使って診断・治療の意思決定を行なう途中で、なぜこのデータが必要なのか(WHY)とか、なぜこの結論になったのか(HOW)などの、いわゆる“推論の道筋”(line of reasoning)を説明する機能を持たせてあることである。これらのアイデアは現在でも極めて重要なものと考えられており、MYCINがとうとう現場で利用されずに終わったにもかかわらず、歴史上のシステムとしてだけでなく、最新のシステムの一つとしても評価できる所以である。

MYCINと同世代のエキスパート・システムには、因果ネットワーク・モデルを用いたCASNETやより一般的な(フレーム・システムに近い)ネットワーク・モデルを用いたPIP、(階層的仮説構造とルールのモジュール化による)いわゆるブラックボード・モデルを用いたHEARSAY IIなどがあるが、いずれも特定の問題のための専用プログラムとして開発された。

70年代後半になって、これらの試を通しての経験が土台となり、汎用パッケージ化されたシステムが開発されるようになった。これらが前述のEMYCINなどのいわゆる知識表現言語(Knowledge Representation Language)あるいは汎用エキスパート・システム開発言語などと呼ばれるプログラムである。

4 汎用言語の意義

汎用言語化されることの利点は極めて大きい。言語あるいはシステムの開発者の立場から見れば、設計のフィロソフィやアイデア等を具現化し、色々なアプリケーションあるいはユーザを通してそれらを評価できる。これは、一種の思考実験の場を提供するという意味を持つ。第2に、只1つのソフトウェアの設計に努力が集中できるから、より能力や品質の高いソフトウェアを開発することが容易となる。第3に、ユーザの経験や要求を基にして、少しずつシステムをグレード・アップすることができる。一般にソフトウェアは、その寿命期間を通じて、作業量の偏りはあるとしても、開発が常にと考えると考えるのが妥当である。これは知識型の汎用ソフトウェアについても同じことが言える。次に、ユーザの立場から考えると、使い易い汎用言語が提供されることによって、自分のアイデアを色々試してみることが容易にできる。もし、従来のプログラミングでこの様なことを行おうとすると、一々開発者(プログラマ)の

手をわずらわせなければならず、試みを中断するかはしよることになる。

但し、このような事ができる為には、理解し易く使い易いインターフェイスが不可欠である。知識型システムに於ては、推論制御機構よりも、知識獲得支援機能（知識ベース・エディタ）の方がはるかに重要であり、その次には推論の説明機能が重要であると思う。これはどのタイプのプログラム言語についても言えることであるが、設計者の意図した能力以上の機能をユーザが引き出してくれるものである。それは、ユーザの努力が知識の構築に集中できることから来ると考えられる。

さて、PSは知識の表現がプロダクション・ルールと呼ばれる、IF-THEN型の単一の形式で表われ、このルールの集合が知識ベースを構成する。一方推論制御は、極めて単純なAND/ORトリートの操作である。従って、知識（ルール）の変更や追加がシステムの挙動や能力に敏感に現われるので、ユーザが試行錯誤的にエキスパート・システムを理解したり、（小規模な）プログラムの開発には、最適な言語の1つであると思う。

知識表現が一様化され、推論機構が単純であると言うことは、設計者から見ると、各種の説明機能や学習機能を実現しやすいという利点がある。事実、これまでに開発されたシステムでこの両方の機能の実現が試みられているのは、PSあるいはルール型システムにかなり偏っている。

但し、このような利点は全て、知識が複雑化し大型化した時には、知識の理解し易さ、システムの実行速度、システムの問題解決力などが低下するという欠点をもたらす。この問題については、ここではふれない。

5 基本的構造

PSは、

IF situation THEN action

IF condition THEN conclusion

等の形式をもつプロダクション・ルールの集合から成る知識ベース（Knowledge Base、KB）、この知識を使って推論を制御するための推論機構（Inference Engine、IE）、および（概念）対象の事実（facts）の集合からなるデータ・ベース（Data Base、DB）で構成されていることは良く知られている通りである。なお、KB、IE、DBをそれぞれルール・ベース、インタープリタ（あるいはモニタ）、グローバル・データベースなどと呼ぶことや、ルールのIF部、THEN部をそれぞれLHS（Left Hand Side）、RHS（Right Hand Side）と呼ぶこともある。また、事実をfactual knowledge、ルールをrule-based knowledge と考えることができ、KBがDBを含むものとも考えることもできる（13）。実際のシステムには、この他に実行時の状態を管理するためのワーキング・メモリも必要となる。また、応用分野によってシステムの構成や推論制御のメカニズムは様々である。ここで対象とする意思決定支援システムの場合には、推論メカニズムはforward chaining、backward chaining、あるいは両者のくみあわせのいずれが多い（後述）。

図1はもっとも基本的なPSにおける推論のメ

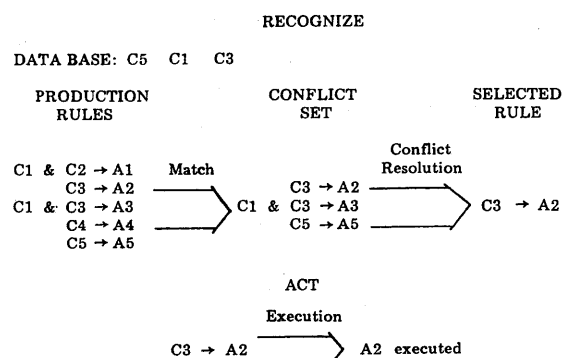


Figure 1. Production system recognize-act cycle

カニズムを示す為の図である(1)。ここでは、DBは単に記号の集りであり、プロダクション・ルールは“条件→ 実行”の形式をしている。このシステムは次の様に動作する。

- 1) プロダクション・ルールのLHSがDB内の事実(記号)と照合され、条件に合格した全てのルールが取り出されてconflict setを構成する。このプロセスをパターン・マッチという。
- 2) Conflict set から1つのルールをある基準に従って選択する。この処理をconflict resolutionという。
- 3) ルールの“実行部”(RHS)が実行され、その結果としてDBが更新される。1へ戻って同様の処理をくり返す。

1、2のプロセスをrecognition、3のプロセスをactionと呼ぶ。従って、PSにおける推論の実行は、“recognize—act cycle”の単純なくり返しである。実行を終了させるルールが“実行”された時、もしくはマッチするルールが無いとき、実行を終了する。このPSは、ルールの条件部をDBの内容と照合して実行部を実行する推論方式をとっており、データ駆動型(data driven)推論と呼ばれる。実際のPSの設計では、conflict resolutionをどうするか、DBの内容や構造をどうするか、ユーザとの質問—応答をどうするか、など色々な問題があり、目的に応じて選択しなければならない。

5. 1 意思決定支援システムとしてのPSの構造

意思決定を図2に示すような簡単なモデルとしよう。すなわち、与えられた事実(facts)に基づいて、結論仮説の中から1つを選択して結論とすることが意思決定の問題である。しかし、一般には事実

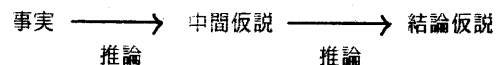


Figure 2 A simplified model for decision making.

から直接結論へ至るのではなく、中間的な概念・事実を経ることが多く、この方が高度な判断に適している。これを中間仮説(intermediate hypotheses)と呼ぶ。

知識とは、ここでは事実および仮説の集合、および判断の為のルールの集合である。ルールをこれらの事実に適用して、事実から結論(goal)に至る処理プロセスを一般に推論と呼んでいる。なお、全てのルールは正しい法則として与えられており、これを適用して特定の事象を説明することが推論であると考え、これは演繹推論に属する。

プロダクション・ルールは全体として、事実と結論を結びつける条件文の集合であり、図3に示すようなAND/ORトリートを構成する。各ノードはRHSであり、そのノードの子供達からのブランチがLHSである。ただし、実際のプロダクション・ルールのLHSはANDとORの組合せが許されることが多いので、1つのルールで

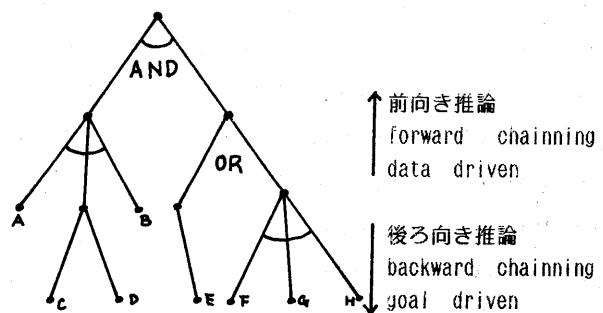


Figure 3 An AND/OR tree.

図3のトリートが表現できる。また、各(結論)仮説に対して1つのトリートが形成されるので、知識は複数個のAND/ORトリートから成る。

5. 2 基本的推論技法

推論の基本的方法は、前向き推論、後向き推論、双方向推論の3つである。

前向き推論： これはforward chaining またはdata drivenなどと呼ばれ、事実あるいはデータから出発し、特定の仮説を選択するやり方である。一般に、この推論では結論に関係の無い部分木が評価されるという無駄が起るが、ユーザとの質問-応答を伴わない自動データ入力方式のシステム環境には適する。EXPERT、OPS、VMなどがこの方式である。

後向き推論： これはbackward chaining またはgoal drivenなどと呼ばれ、goalからスタートし、その仮説が適当か否かを立証するまで逆方向に推論を進める。Goal としての仮説を与えて推論が開始され、その仮説をRHS に持つルールが知識ベースから選択され、そのLHSが評価される。もしDB中の事実で評価できないときは、これをsubgoal として、知識ベース中の他のルールを使ってこのsubgoal を評価するという必要をただくり返す。従ってこれは再帰的手続である。この推論では、結論に関係のある部分木のみが評価されるので効率が良いが、なかなか否定も肯定もできないときは無駄に木を生成するという問題がある。他の方法で見込みの高い仮説を前もってしることができるときには利点が生じる。MYCIN (EMYCIN) がこの方式である。

双方向推論： 与えられたデータに基づいて前向きに推論を開始し、これによって得られた仮説を結論として採用するか否かの決定を、この仮説をgoalとして後向きに推論するというやり方がこれである。システム制御や、少い情報に基く推論のスタートには、自然である。我々が日常行っている判断には、行ったり来たり“のどうどう巡り”が少くない。“仮説生成と検証”(hypothesize and test) が最も自然で柔軟な推論のやり方であり、双方向推論はこのアイデアを取り入れている。ただし、本格的には、フレーム・モデルやブラックボード・モデルが必要である。RITAはこの方式のPSである。

以下、PSの各機能について考察しよう。

6 データベース

DBは、推論に直接関係する事実および仮説を管理する機能であり、推論の結果はここに反映される。DBの内容および構造の設計は、PSの性格や能力に大きく影響を与える。

最も簡単なDBは、“object-value”組の単なる集合である。例えば、社員、給料、病気等のobjects が“YAMADA TARO”、200000、“YES”等のvaluesをとる。教育用システムにはこの程度のDBでもよいであろう。

次のレベルのDBは、EMYCINやRITAが採用している“object-attribute-value”組の集合である。1つのobjectは複数個のattributesをとることができ、各attribute は1つ（または複数）のvalueをとることができる。例えば、医学検査というobjectに血圧、血液型、身長、光反応のattributesを与えると、80-130、“A”、“Unknown”、“+”などのvalueをとる。Object の概念を大きくするか小さくするかによって、ユーザはある程度PSの性質を制御できる。

DBを構造化することができる。例えばEMYCINでは、object (context と読んでいる) の階層構造をcontext treeとして定義することができる。このtreeとルールとの関係は明示的に定義することができるので、知識の構造化・モジュール化をある程度実現している。EMYCINによるエキスパート・システムの設計において、context treeの設計が最も重要であると言われるのは納得できる。

DBの構造化を更に進めることができる。仮説を階層化し、各レベルに複数のobject-value 組を割当てるように

したものが、いわゆるブラックボード・モデルである。このモデルは更に、仮説レベル間にルールをグループ化したノーレッジ・ソースを割り当てることによって、高度なモジュール化と推論制御を実現している。セマンティック・ネットワークやフレーム・システムによってDBを実現すれば、更に強力な表現が可能となる。但し、この場合にはPSと考えるよりも別のシステムと考える方が自然である。Attached proceduresを許せば、PSとしての色彩はほとんど消えてしまう。

7 プロダクション・ルール

LHSは、DB内のobject-attribute-value組に対する論理演算であり、結果として論理値を取る。最も簡単な場合には、内部表現が

(EQUAL A "B") あるいは

(AND (EQUAL A "B") (LESS-THAN C 100))

である。上の例は、“もし A=“B” ならば”であり、下の例は、“もし A=“B” かつ C< 100 ならば”を表わす。AND、OR、NOTを何重にも許すようにすれば、1つのLHSで複雑な条件を表現することができる。

RHSは、LHSの条件が満たされたとき実行すべきactionsを指定したものである。実際には、前もって準備された関数や手続きの起動を指定する。DBの内容の更新が主であるが、ユーザへのメッセージの出力、推論制御の変更なども指定できる。DBの更新は、goal-driven型推論では1つに限定しなければ混乱が生ずるが、data-driven型では複数個あってもかまわない。

LHSおよびRHSにユーザ指定の関数や手続きを許すようにすれば、PSの柔軟性はかなり高められる。また、前述のノーレッジ・ソースの技法を採用すれば、知識ベースの見通しが良くなるばかりでなく、実行効率も向上する。

8 Conflict resolution

PSの基本的推論機構については5. 2で述べたが、実際の推論制御機構を設計するには色々検討すべき問題がある。その中でconflict resolutionについて簡単に考察する。Conflict resolutionは、パターン・マッチによって2つ以上のルールが候補となったとき、ある基準で1つを選択する処理である。これには次の様にいくつかのものがある(8)。

ルールの優先順位： 各ルールにselection criteria用の優先順位を与えておく方法である。

ファースト・マッチ： KB内を定義されている順序でテストしていき、最初にマッチしたルールを選択する。ユーザが指定できるので簡単かつ有用である。

最近使用のルール： 最も近い過去に実行されたルールを選択する。この逆もある。時間はcycle 数で計る。

条件の指定： 各ルールと対応するDB内のobjectに条件が指定される。

類似度による選択： 前に実行されたルールのLHSまたはRHSとの類似度がテストされ、最も高いもの、または最も低いものが選択される。

ランダム・セレクション： 乱数を使って選択する。

この他に1つのルールは只1度だけ実行が許されるようにしたり (once only)、最も複雑なルールを選択する方法、

agendaと呼ばれる以前に登録されたルールの中からある基準で選択する方法なども考えられる。

Conflict resolutionは、システムの性格に影響を及ぼす重要な機能である。システム設計の段階で推論制御機構の中に組み込んでしまうか、メタ知識としてユーザが指定できるようにしなければならないが、default として標準のものを準備しておき、ユーザも指定できるように設計すれば柔軟性が高くなる。また、data-drivenとgoal-drivenとは異った機能が要求される。因みに、EMYCINではマッチした全てのルールが実行（評価）される。これはCFの効果が最後まで解らないからである。

9 システムの開発について

典型的汎用PSは、KB（または、ルール・ベース）、DB、ワーキング・メモリ、モニタ、説明モジュール、KBエディタおよびユーザ・インターフェイスから構成される。説明モジュールは前述の“推論の道筋”をユーザに説明するための機能であり、KBエディタは、エキスパートからの知識獲得支援を行なうための機能であり、KBとDBの構築および管理に用いられる。ユーザ・インターフェイスは、質問-応答の管理やルールの自然言語による外部表現と内部表現との自動変換などを行なう機能である。

PSは、KB、DB、ワーキング・メモリおよびモニタがあれば一応動くが、残りの3つが欠けたシステムは使いものにならないことは明らかである。KB、DBおよびワーキング・メモリはデータ構造であるから、プログラムの作成とは異ったものである。以上を念頭に置いて、インプリメンテーションの順序について検討しよう。

第一に開発すべきものはKBエディタである。これができれば、KBやDBの定義、変更、確認などが極めて容易となり、残りのモジュールの開発が格段に促進される。つまり、KB、エディタの開発は知識型システムのインプリメンテーション環境の整備としての意味を持つ。次に開発すべきものは、ユーザ・インターフェイスである。これによって開発環境は更に大きく改善される。その次がモニタである。ここまでくれば、PSとしてはほぼ完全な利用ができる。そして最後が説明モジュールである。我々の経験によると最後の説明モジュールが整備されると、ドメイン・ユーザの反応は極めて大きく、システムの能力がはじめて評価できるようになる。ユーザが使い始めると色々な要求がよせられる。これをシステムの改良に反映させることができる。と同時に、システムの限界や欠点も明らかとなるので、新しいより汎用なモデルへのヒントとなる。

まとめ

PSの開発について問題点を整理したが、紙面の制約等で十分に議論できなかった。PSの特徴や本質的な制限など、ここで触れられなかった問題は別の機会にゆずる。また、SMARTおよび、TRYはバージョン0がほぼ完成しているので、これについても別の機会に報告する予定である。

謝辞 UTILISPの使用を快諾していただいた東京大学和田研究室に深謝する。現在、FMSとSMARTの開発に使っているが十分な能力を持っている。特に、実行速度は特筆すべきものがある。米国のINTERLISPやMACLISPに対抗できると思うので、利用環境を十分なレベルまでぜひ整備していただきたい。

References

- 1) Waterman, D. A. and Frederic Hayes-Roth : An Overview of Pattern Directed

- Inference Systems, Pattern Directed Inference Systems, Academic Press, pp 3-22, 1978
- 2) Buchanan, B. G. and R. O. Duda : Principles of Rule -Based Expert Systems, HPP-82-14, Stanford University, 1982
 - 3) Davis, R. and J. King : An Overview of Production Systems, Machine Intelligence 8, Wiley, pp 300- 322, 1976
 - 4) Frederic Hayes-Roth, Principles of Pattern-Directed Inference Systems, Pattern Directed Inference Systems, Academic Press, pp577 - 601, 1978
 - 5) Fagan, L. M., E. H. Shortliffe : Computer Based Medical Decision Making : From MYCIN to VM, HPP-79-18, Stanford University, 1979
 - 6) Scott, A. C., R. Davis and W. Clancey and E. Shortliffe : Explanation Capabilities of Production -Based Consultation Systems, STAN-77-593, 1977
 - 7) Forgy, C. and J. Mc Dermott : OPS, A Domain -Independent Production System Language, IJCAI-5, pp933 - 939, 1977
 - 8) Mc Dermott, J. and C. Forgy : Production System Conflict Resolution Strategies, Pattern-Directed Inference Systems, Academic Press, pp177 - 199, 1978
 - 9) 上野晴樹、伊藤秀昭 : FMS : フレーム理論に基づく汎用知識表現言語, AL82-64, 電子通信学会, 1982
 - 10) 上野晴樹 : COMEX : 汎用知識型エキスパート・システム開発言語、知識工学と人工知能 28-5, 1982
 - 11) Chikayama, T. : UTILISP Manual, METR 81-6, University of Tokyo, 1981
 - 12) Buchanan, B. G. : Research on Expert Systems, HPP-81-1, 1981
 - 13) Van Melle, W., A. C. Scott, J. S. Bennett and M. Peairs : The Emycin Manual, HPP -81-16, 1981
 - 14) 上野晴樹 : COMEXマニュアル、東京電機大学経営工学科, 1982
 - 15) Post, E. : Formal Reductions of the General Combinatorial Problem, Am Jnl Math, Vol65, pp197-268, 1943
 - 16) Minsky, M. : Computation: Finite and Infinite Machines, Prentice Hall, 1967
 - 17) 渡辺正信 : 概念ネットワークを利用した知識獲得支援、情報処理学会知識工学と人工知能 28-6, 1982
 - 18) Suwa, M., A. C. Scott and E. H. Shortliffe : An Approach to Verifying Completeness and Consistency in a Rule -Based Expert System, HPP-81-5, Stanford University, 1981