知識工学と人工知能　54－5
（１９８７・９・１８）

ＯＰＳ５のプログラミングパラダイムを使った生産計画シュミレーション

山岡喜代司
日立造船情報システム（株）　開発本部

知識工学手法を使ったアプリケーションは故障診断、医療診断等の診断型、質量・成分分析等の分析型、あるいはコンピュータシステム構成等のコンサルテーション型が大部分をしめている。知識工学がこうした分野に適用されるのは、その手法がアプリケーションの内容とよく整合するためであると解釈されるし、適用されるべき最適な分野であるとも理解される。

本報では知識工学をより一般的なアプリケーションに適用するために計画型の範ちゅうに入る生産計画シュミレーションを例としてとりあげ、知識工学手法を利用し応用することを試みた。知識工学手法はＯＰＳ５のもつプログラミングパラダイムであるトリガーによるプログラミングを使い、生産計画の主となる概念である生産のフロー、生産の時間概念、ルール項目及び状況変化を具現した。

The Production Planning Simulation Applying to the Programming Paradigm of OPS5

Kiyoshi YAMAOKA

Development Div., Hitachi Zosen Information Systems Co.,Ltd.
1-4-10,Takanawa,Minato-ku,Tokyo,108 Japan

The application programs to which the techniques of knowledge engineering have been applied are becoming increasingly available. The majority of these programs fall into the most popular fields of the expert systems classification, fields such as diagnosis, interpretation, and consultation. The reason for this may be that programs in these fields are understood to be the most suitable for the application of knowledge engineering.

Here, the production planning simulation is selected as an example in order to apply the technique for broad, general applicatin programs. It's chief concepts, the flow of the production process, the time in the process and the rule item and transition of the situation are implemented with the techniques embedded in the programming paradigm of OPS5, typically programming by trigger.

１．はじめに

知識工学を利用したアプリケーションの多くは診断型、分析型、コンサルテーション型のものなどで、多数発表され実用として使用され成果をあげているものが多々ある。このことはある状況が与えられたとき、それと合致する状況が関連する状況をつくり出し、それが次々と連鎖し最終の状況に致り結果が得られるといった過程がアプリケーションの内容と知識工学手法によく合致していることからくると思われる。多数のものから要求に合うものを選び出すことが単にデータベースの検索でないことはその間に種々の条件等が介在することにあり、知識工学手法が取り込まれるのもその理由による。比較的発表される数の少ない設計・計画型のアプリケーションも内容的にみると設計支援のためのコンサルテーション等で同様の部分を扱っているものが多くみうけられ、その部分こそが現時点で実用に供するために知識工学手法を導入するべきところなのかもしれない。

本報においては知識工学手法自体を使用するだけでなく、また上述のその部分に適用するのでもなく、その手法を利用し応用してより一般的なアプリケーションを構築することを計画型の範ちゅうに入る生産計画シュミレーションを例としてとりあげ試みた。

２．生産計画の概要

本生産計画は一つの工場から他の工場への部品の受け渡しの計画を行なうものであり、その概要は図１のようになる。一つの工場をオ１工場、他の工場をオ２工場とすると、まず幾種類かの部品がオ１工場の製造機械で生産されることから始まる。各部品の１部品当りの生産時間（分／個）と部品の生産順序は決められており、各部品の種類をＡ、Ｂ、Ｃ、・・・とすると生産順序はＡ→Ｂ→Ｃ→・・・のようになり同じ種類のものが再び現われるＡ→Ｂ→Ｃ→・・・→Ａ→Ｂ→Ｈ→・・・のようなケースも起りうる。生産順序に従いオ１工場でつくられる各部品はオ１工場にあるストックステージに順次仮置きされるが、その数は最大数による制限をうける。ストックステージにたまった部品は定期的に運搬され、オ２工場の部品の種類別にあるストックステージに対応する部品ごとに仮置きされる。一回の運搬で運搬可能な部品の最大個数は決められており、その個数をこえることはできない。

オ２工場にある部品別のストックステージはそれぞれに最大数が決められており、それらの部品を一定の時間間隔で取り出し組立機械により製品として完成させる。各部品をストックステージより取り出し使用する１部品当りの使用時間（分／個）は決められており、オ１工場での部品の製造及びその運搬とかかわりなくストックステージより取り出され使用されていくため、常にストックステージに部品を供給しておく必要がある。

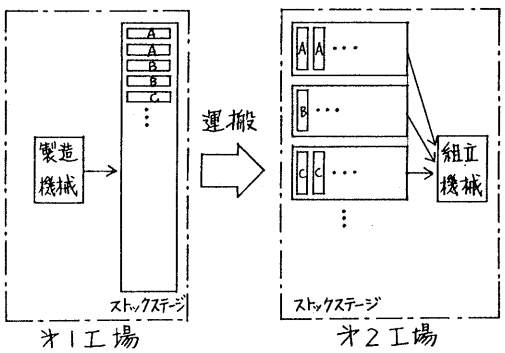

図１

オ１工場における各部品の生産個数はオ１、２工場にあるストックステージのあき具合により決定される。オ１工場よりオ２工場へ

の各部品の運搬個数は第1工場にあるストックステージに部品が存在し、かつ第2工場にあるストックステージにあきスペースがあり、かつそれぞれの部品の合計が運搬可能最大個数をこえていないことにより決定される。

3．生産計画のシュミレーション

シュミレーションの目的はある条件を設定したとき、それにより内容がどのように変化していくかを知ることにある。内容の変化をみながら最適の条件をさがしだすのが一般的であるが、さらにすすめば自動的に最適条件を導き出すこととなる。

本生産計画のシュミレーションは前者の段階に位置し、設定された条件に対して内容が時間経過と共にどのように変化していくかを見ていくこととなる。設定される条件は各部品の一日当りの生産個数である。生産個数が決まると各部品の生産時間と生産順序に従い順次部品がつくられてゆき、第1工場にあるストックステージに仮置きされる。内容の変化をみる第1は第1工場にあるストックステージにおかれている部品の個数とその種類である。第1工場にあるストックステージにある部品は第2工場での部品の使用状況による第2工場にあるストックステージのあき具合により、第1工場側のストックステージから第2工場側のストックステージに運搬される。内容の変化をみる第2は第1工場から第2工場へ運搬される部品の同じく個数とその種類である。第2工場にあるストックステージに仮置きされた部品は各部品の使用時間に従いストックステージから取り出され、組立機械により製品として完成される。内容の変化をみる第3は、第2工場にあるストックステージは各部品ごとに分けられているため、各々のストックステージにある部品の個数である。第1から第3までの内容は時間の経過と共に刻々と変わるが目的は当然、時間と共にどのように変化していくかである。

シュミレーションは一日だけでなく数日にわたることもあり、各日の各部品の生産個数及び使用個数は条件として与えられる。計画生産個数は当日の作業が終了した時点で翌日の作業のために第2工場にあるストックステージに適度の部品を残して当日に消費されるように定められなければならない。計画使用個数は実質的には計画生産個数に従属した値となり、同様に当日に消費されるように定められなければならない。計画生産個数と使用個数及び各日におけるそれぞれの個数間に大きく変動があるなら、部品の生産順序等を変化させてシュミレーションを繰り返すことにより均等化された個数にもっていくことも可能となる。

シュミレーションはまず生産個数及び使用個数を既知として開始するが可変的に扱えるのは生産個数及び使用個数とともに、生産順序、第1、2工場側のストックステージの部品の仮置き最大数、運搬時の運搬可能最大数と運搬回数及び各部品の生産、使用時間があるので、これらを考慮して意図する最適のものにむけて試行錯誤することが可能である。

4．OPS5のプログラミングパラダイムの利用

OPS5はエキスパートシステム開発用ツールとして広く使用されており前向き推論を行なうプロダクションシステムルールを提供する。OPS5の長所はそれがもつプログラミングパラダイムである「トリガーによるプログラミング」であるといわれる[1]。トリガーはOPS5の二つの戦略（競合解消）であるL

ＬＥＸとＭＥＡに起因している。競合解消戦略は、

（１）　一度選択されたinstantiationを競合集合から除く。

（２）　より新しいWorking Memory Element（以下ＷＭＥという）を含むinstantiationを選ぶ。

（３）　より条件の厳しいinstantiationを選ぶ。すなわち、プロダクションの条件部で行なわれる値のチェックの回数が多いほうを選ぶ。

（４）　以上で決まらなかった場合は任意のinstantiationを選ぶ。

であり(1)から(4)の順に適用される。ＬＥＸとＭＥＡでは戦略の才２番目のWorking Memory（以下ＷＭという）内にＷＭＥが作られた時刻の新しさであるrecencyの扱いだけが少し異なるが、この相違は実際上大きな意味をもち全く別のルールが発火されることがある。

ＯＰＳ５の利用はこの戦略の内容を使うことにある。戦略の才１番目は既にルールの発火に使われたinstantiationを対象から外すため、一度発火したルールはＷＭ内に変化がなければ再び発火することはない。これを逆にとらえるとＷＭ内に変化を起こせば再び同じルールが発火することになる。トップレベルのREMOVEとMAKE、あるいはMODIFYによりＷＭ内に要素を消滅→生成することで再び同じルールが適用可能となる。すなわち、ルールの条件部にマッチングする特定のＷＭＥを消滅→生成することで任意の時点で特定のルールを再び発火させることができる。ＷＭＥをあたかもトークンのように使い、トークンがＷＭに現われると特定のルールが発火しそれにより次々とルールが発火していく、再び同じルールを発火させようとすれば取除いておいたトークンを新たに投入する、逆に特定のルールの発火をさけるためにＷＭに存在するトークンを取り除くということを行なうことにより制御が可能となる。この方法はルール内に陽に表現することができるため、プログラム上での判読が容易である。

戦略の才２番目はＬＥＸとＭＥＡでは少し異なるが、最新のＷＭＥが選ばれることである。ＷＭＥは常に生成・消滅を繰り返し、大きなプログラムになればその頻度は相当なものになると推測される。それを利用することはＷＭＥの生成・消滅を把握し、かつそのことによる新しさの順位を知っていなければならず、このことはルールがふえると非常に煩雑で事実上不可能なこととなる。さらにルールにおいてＷＭ内の様子を陽に表現出来ず暗にそのことを表わすことになるためプログラム上で簡単に判読出来ない点もあり、この戦略は利用という観点からみると扱いにくい。

戦略の才３番目は条件部の厳しいルールが選ばれることである。ルールを特定の順番で発火させていくとき最初に発火したいものの条件部を厳しくしておき徐々にゆるめていくと、それに従ってルールが発火されることとなる。プログラム上もこのことを一目瞭然ととらえることができ判読も容易である。

５．シュミレーションの構築

生産計画プログラムを構築しシュミレーションを行なうには、生産のフロー、生産の時間概念、ルールとして抽出する項目及び時々刻々と変化する状況のそれぞれの扱いが焦点となる。以下にＯＰＳ５のプログラミングパラダイムの利用の範ちゅうにおけるそれぞれについての考え方を説明する。

５－１．生産のフローの表現

フローは大別すると二つに分けられる。一つはシュミレーションの対象となるフローで変更が起りうることを予定するもの、他の一つはシュミレーションの対象とならないフローで固定化されるものである。

変更が起りうるフローは変更の簡便化に対応するため、明示的かつ独立的に表現されなければならない。ＯＰＳ５はその記述をすべてルールという形をとるため、一つのルール内で表現しておけば独立的に記述でき、しかも明示的となる。

ﾈｲﾙ工場での生産順序は特定のルールでもって、

```
(作業位置  ∧生産位置  1
           ∧作業順序  A  B  C ・・・          )
```

というＷＭＥをつくり、生産作業が行なわれる順番に部品の種類を'作業順序'に列挙するとともに現在作業されている作業順序上の位置を'生産位置'に保持する。

```
(Ｐ  作業順序  A →B →C→ ・・・
   {(計算作業  ∧生産物種類  <>H
               ∧内容        生産量)   <計算作業>}
  ⋮
⟶ ⋮
   (modify  <計算作業>   ∧生産物種類  (substr・・・   ))
   (modify  <作業位置>   ∧生産位置    (compute <ccp>+1))
   ⋮                                              )
```

のルールが発火されるごとにＷＭＥ内の'作業順序'より現'生産位置'の次の部品の種類をとり出し、'生産位置'のカウンターを一つ進め、その部品の種類でもって別のルールに起動をかける。したがって生産順序の変更は'作業順序'の内容を変更すればよいだけのこととなり、明示的かつ独立的とした意図を満すこととなる。

固定化されるフローはルール内に以下の方法でもってちりばめることができ、これらの組み合わせにより自在にフローをつくり出すことができる。

A．ルールは条件部の各条件要素の各属性のもつ属性値がすべて合致しないと起動しないため、特定の属性に着目してその属性値を変化させる。変化した属性値に合致するルールが次に起動されることとなるので、そのルール内で再び同じ属性の属性値を変化させると、さらにその条件に合致するルールが起動される。このことを次々と繰り返すことによりフローがつくり出される。（図２）

B．ルールの条件部にある(ない)条件要素を消し(つくり)、そのことにより次のルールの起動を制御する。条件要素があたかもトークンのイメージでＷＭ内に生成・消滅し、トークンが生成したとき合致するルールが起動され、トークンが消

```
(Ｐ
   {(       ∧
            ∧ STATUS  PATH_1) <ΔΔ>}
  ⋮
⟶ ⋮
   (modify <ΔΔ>  ∧STATUS PATH_2)
   ⋮                                )

(Ｐ
   {(       ∧
            ∧ STATUS  PATH_2) <ΔΔ>}
  ⋮
⟶ ⋮
   (modify <ΔΔ>  ∧STATUS PATH_3)
   ⋮                                )
```

図２

```
(Ｐ
⟶(                )
  (make                ))
```

滅すればその瞬間に停止することとなる。フローの続行、中断を意図的に任意につくり出すことが可能となる。（図３）

```
(P
  -->{(                ) <Δ Δ>}
     (remove <ΔΔ>              ))
```

図３

C．ルールの条件部の条件要素数をかえることでルールの起動される順番をつくり出すことができる。同じレベルにあるルールであるが、ある特定のルールが最初に起動され、それが合致しなければ次の特定のルールが起動される状況がつくられる。ＯＰＳ５はＩＦ　ＴＨＥＮ形式で記述されるが、この方法を使うことで二つのルールでもってあたかもＩＦ ＴＨＥＮ ＥＬＳＥをつくることが可能となる。（図４）

```
(P
   (L1          )
   (L2          )
   (L3          )
-->  ⋮           )
```

```
(P
   (L1          )
   (L2          )
-->  ⋮           )
```

図４

これらの方法は変更が起りうるフローにもちいた方法ほど明示ではないにしてもルールを判読すれば理解できるかたちとなる。

フローをつくり出すさらに別の方法はＯＰＳ５の戦略であるより新しいＷＭＥを含むinstantiationを選ぶ方法を利用することであるが、この方法をもちいるとＷＭＥの新しさ、すなわちＷＭＥに付くタイムタグを常に意識していなければならない。ＯＰＳ５の利用で述べたようにルールの数が少なく単純であればタイムタグを考慮することはできるが、複雑になるとまず不可能であること、タイムタグ自体が内在的存在でありルール内のどこにも現われないことから、この戦略を利用してフローを作り出すことは考えなかった。

５－２．生産の時間概念の表現

時間は連続的に経過して行く。そのような時間を連続量に近い微少時間の集合として扱うのでなく、ある時間間隔における時間断面として断続量的に扱う。

時間に関係するのは才１工場での各部品ごとの生産時間（分/個）、才２工場での各部品ごとの使用時間（分/個）および才１工場から才２工場への部品の運搬時刻である。才１工場では生産順序および生産時間に従い各部品は生産され才１工場にあるストックステージにおかれ、才２工場ではそこにあるストックステージより各部品がとり出され、組み立てられ製品として完成していく。この流れは連続的であり、どの段階においても特別に着目すべき時点はなくすべて平等である。したがって着目するとなるとと才１工場から才２工場への部品の運搬時刻となり、その時刻で才１工場にあるストックステージにどの種類の部品が何個、才２工場にある各部品のストックステージにそれぞれ何個あるかを明確にし、どの部品を何個運搬すればよいかを判断しなければならないという内部的要求にも合致することになる。運搬時刻をルールの条件要素の一つとして取り入れ、最初の運搬時刻までの才１、２工場での状況を計算する。その時刻での状況が判明すると運搬時刻を更新して再び同じ過程のルールを起動し次の時刻までの状況を計算する。これを繰り返すことにより時間の経過と共に進行していく生産状況を把握し生産推進に起動をかけていく。時間概念はこのように時間断面でとらえられ、かつそ

れは内部的要求とシュミレーションとして必要とされる特定時間局面での内容変化の要求とを同時に満すこととなる。

5－3．ルール項目と状況変化の表現

状況と共に変化していく内容をWMに、状況が変化しても変わらない内容をProduction Memory（以TPMという）に割り当てるのがOPS5の一般的な使い方であるといわれ、この場合もその考え方に無理なくあてはめることができる〔2〕。したがってPMに、

- 才1工場での生産量の計算
- 才2工場での使用量の計算
- 才1工場から才2工場への運搬量の計算
- 作業順序の設定

をルールとして割り当てる。これらのルールは本生産計画内では普遍であり、動的に変わるものではない。

WMには部品の各種類ごとにその関係する項目が推移していく状況を図5のような表イメージで割り当てる。関係する項目は才1、2工場にあるストックステージ上の部品の個数、才1工場で1日に生産する及び才2工場で1日に組み立てのために使用する部品の量である。才1工場で1日に生産する部品量は部品が生産されるごとにその部品量から減算され、部品量が0となればそれ以上生産出来ないこととなり、それに相当分が才1工場側のストックステージに加算されることとなる。才2工場で1日に組み立てのために使用する部品量は同様に使用されるごとにその部品量から減算され、部品量が0となればそれ以上使用出来ないこととなり、それに相当分が才2工場にあるストックステージより同時に減算されることとなる。推移変化は運搬時にも発生し、才1工場側のストックステージより搬出された分が才2工場にあるストックステージに加算される。

部品の各々にはこのようにして変化していく項目とともに各部品に特有の固定した値をもつ項目、たとえば才1工場での生産時間、才2工場での使用時間等、及び変化していく途中に一時的に必要となる項目とがある。これらの項目も同様に扱うと便利であるため、表イメージに組み込みWM内におく。したがってWM内をみることにより各部品ごとの特定時間における変化状況を一目瞭然にとらえることができ、また途中状況の一時記録につかっている項目に着目すると、それを参照することにより異常状況が発生しているか否かのデバッグに利用出来る。

図5

部品種類	才1工場側ストックステージ	才2工場側ストックステージ	才1工場生産量	才2工場使用量	才1工場生産時間	才2工場使用時間	
A							
B							
C							
⋮							

5－4．構築結果

1日のシュミレーションを21ルールで記述することができた。ルールは初期設定、才1工場での生産量の計算、才2工場での使用量の計算、才1工場から才2工場への運搬量の計算及び作業順序の設定に大別される結果となった。

初期設定のカテゴリーでは前述のように状況の変化を扱うための各部品に個有な値等をもった表イメージのデータをWM内につくり出す。オ1工場での生産量の計算、オ2工場での使用量の計算及びオ1工場からオ2工場への運搬量の計算のカテゴリーではそれぞれの計算をルールとして表現しているが、1ルールで処理出来ないため数ルールにて記述している。数ルールに別けたため、その適用順序が問題となるが、それに対してはOPS5の戦略を利用してフローをつくりルールが順次に適用されるようにつくりあげている。作業順序のカテゴリーはこのアプリケーションのメインフローをつくりあげており、フローを変更するときはこのカテゴリー内にあるルールを変更すればよいこととなっている。

シュミレーションが一日で終了しないときは次の日のシュミレーションを開始するというルールを追加し、そのルール内でその日に必要となる値を表イメージのデータに追加していけばよく、必要となる日数分だけルールを追加することとなる。したがって日数がふえるとそれに応じてルールを増やせばよく容易に拡張していけることとなっている。インプリメントはCommon Lisp環境下で動作するOPS5をもちいて行なった。

6．おわりに

本報では知識工学手法を利用し応用した生産計画のアプリケーションの構築について述べた。ここで提示した手法は生産計画のアプリケーションにとどまらず、あらゆる分野のアプリケーションに応用し適用できると考えられる。各分野ではその分野のキーとなる概念をどのように扱うかが、この生産計画のアプリケーションで生産のフロー、生産の時間概念、及び時々刻々と変化する状況の扱い方で述べたと同様に具体的な実現方法として問題になってくるが、それらは一般化できるものではないので個別に解決されなければならない。しかしながら同様の概念には同様の手法が適用出来ることを考えると、アプリケーションは際限なくあるとしてもすべて最初からではなく、その手法は累積していけるため発散の方向に向わず収束していけると考えられる。

種々のアプリケーションが知識工学手法で記述できるとしたとき、既存の手法との比較評価が生じてくる。今までの手法がアプリケーションの実行手続を記述したものといえども自動的に処理を行なうアプリケーションであればあるほどその中に知識といえるものが内在するわけであり、すでに知識が存在しているとさえいえる。ただその知識が散在しているのが通常であるかもしれないが、まとめようによっては知識を集中的に管理することも可能である。こうなると知識工学手法がすでに内在していたといっても過言でない。

本報においてこの問題に答えるのはこの一例だけでは不十分であると考えている。生産計画のアプリケーション構築はそうした問題を解くために手近かに選んだ例であり、比較評価への方向の第一歩である。種々のアプリケーションに知識工学手法を適用していったとき、この手法の適切な比較評価の説明が得られてくるのではと考えている。

参考文献

[1] 戸沢："プログラミング言語としてのOPS5"、知識工学と人工知能研究会、45－9、1986

[2] L. Brownston 他："Programming Expert System in OPS5"、Addison-Wesley