知識工学と人工知能　59－19
（１９８８．７．７）

プロダクションシステムにおける最大・最小値の取扱いとリアルタイム機能への応用

竹中　一起、横井　玉雄
住友金属工業㈱　システムエンジニアリング本部

プロダクションシステムの枠組の中で、対象データの最大値とか最小値を扱う場合、高速のパターンマッチング手法であるReteアルゴリズムを採用したとしても、データの選出にはかなりの計算時間を要していた。本稿では、最大・最小値の取扱いを効率化するために、Reteネットワーク内でＷＭＥを順序列として管理する方式を提案し、Reteネットワーク上での実現方法を示すとともにその効果について述べる。さらに本方式を拡張すれば、サンプリングや時間概念を有するデータについても、Reteネットワーク内で簡潔にかつ高速に処理できることを示す。

Effective Approach to Maximun/Minimun Data in Production System and Application to Realtime Environment (in Japanese)

Kazuki TAKENAKA and Tamao YOKOI
Sumitomo Metal Industries, Ltd.
1-3, Nishinagasu-hondori, Amagasaki, Hyogo, 660, Japan

In the production system, it takes so much time to deal with the maximum or minimum data of objects, even if Rete match algorithm is adopted. In this paper, we propose a new method by which WMEs are managed orderly in Rete network, in order to process those data efficiently. Then we show how to implement the method on Rete network, and discuss the effect. Finally we show that the method can be successfully applied to sampling or time-based data.

Ⅰ．緒言

手続き型言語によるコンピュータシステムの限界を打ち破るものとして、ＡＩ手法、特にエキスパートシステムの枠組が注目されている。我々もプロセス制御システムのより一層の高度化・知的化を図るべく、早くからこの技術に着目し、プロセス制御用計算機上で稼働するエキスパートシステム構築ツールＭＡＲＫＳ－Ⅱ[1]を独自に開発し、社内応用システムへの適用を進めてきた。

ＭＡＲＫＳ－Ⅱでは、推論方式としてプロダクション型と分類型の２種類の推論エンジンを備えている。プロダクション型についてはReteアルゴリズム[2]を採用して、パターンマッチング処理を効率化し、さらに数々の独自の高速化手法[3]を駆使することにより、実時間制御に適用するに充分な高速性を実現している。

ところで、特に設計や計画問題に関する知識を整理してみると、例えば「重量比が最小の材料に関して、含有成分が最大の元素が◇◇であって、その含有量が△△であれば、－－－。」というように、ある要素の値が最大とか最小とかを扱う知識が非常に多いのに気付く。ところがReteアルゴリズムでは推論データであるワーキングメモリ・エレメント（WME：ＭＡＲＫＳ－Ⅱでは「知識データオブジェクト」と呼ぶ）を順序列として管理していないため、最大データや最小データの選出にはかなりの計算時間を要していた。そこで最大・最小の取扱いを効率化することを目的として、Reteネットワーク内でWMEを順序列として管理する方式を提案する。

本稿では、まず今回の提案のベースとなるReteアルゴリズムについて簡単にレビューした後、最大・最小の取扱いについてReteアルゴリズムに内在する問題点を指摘する。次に今回提案する新しいWMEの管理方式とマッチング・アルゴリズムについて述べ、Reteネットワーク上での実現方法を示すとともにその効果について評価する。そして最後にリアルタイム機能への応用として、本方式を拡張すれば、サンプリングや時間概念を有するデータについても、Reteネット上で簡潔にかつ高速に処理できることを示す。

Ⅱ．Reteアルゴリズム

Reteアルゴリズムとは、１回のルール実行によって変化するWMEの数は通常極く少数であること、従って条件部の成立可否をチェックし直す必要のあるルール数はそう多くないことに着目し、変化したWMEに関係するルールだけにマッチング処理を限定することにより推論を効率化することを狙ったものである。さらに複数のルール内に現れる同一固定値のチェックを融合し、また前回までのルール実行で部分的にマッチングしている情報を保持しておくことによってマッチング処理の回数を激減させている。

具体的には知識ベースの全ルールを、図１に示すようなネットワークに展開し、WMEを「トークン」という形でネットワークに流すことによりルールのマッチング処理を行っている。このときトークンの頭に、データ生成のときは＋、消去の時は－の符号をつけてネットワークに流す。

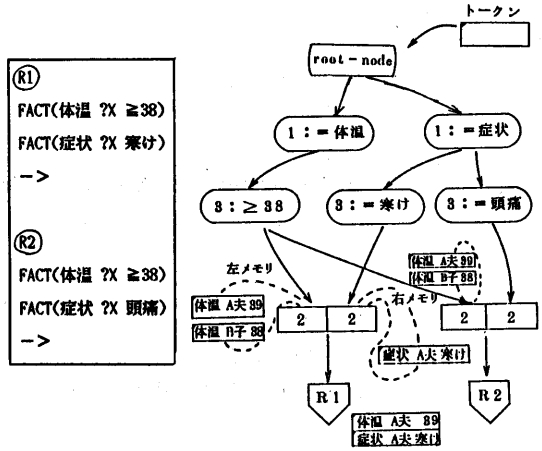


図１．Ｒｅｔｅネットワーク

表１．Ｒｅｔｅネットワークのノード

名　称	ﾙｰﾙとの対応	主　な　処　理
ﾙｰﾄ・ﾉｰﾄﾞ	ﾙｰﾙの集合 （知識ﾕﾆｯﾄ）	ﾙｰﾙの実行部を「ﾄｰｸﾝ」という型にして、全ての子ﾉｰﾄﾞに流す。
ｲﾝﾄﾗ・ﾉｰﾄﾞ	ﾊﾟﾀｰﾝ内の 固定ﾃﾞｰﾀのﾁｪｯｸ	流れてきたﾄｰｸﾝが 条件を満たすとき　下流にﾄｰｸﾝを流す。 条件を満たさないとき　ﾄｰｸﾝを消す。
ｲﾝﾀｰ・ﾉｰﾄﾞ	ﾊﾟﾀｰﾝ間のﾁｪｯｸ ・?変数のﾊﾞｲﾝﾄﾞ ・ﾊﾟﾀｰﾝ間での一致ﾁｪｯｸ （ﾙｰﾙのﾊﾟﾀｰﾝ毎に1個）	①流れてきたﾄｰｸﾝを、その方向に応じて右ﾒﾓﾘ or 左ﾒﾓﾘに格納する。 ②反対側のﾒﾓﾘ内(右から流れてきた場合は左ﾒﾓﾘを探す)に格納されているﾄｰｸﾝと条件のﾁｪｯｸを行い、満足していれば2つのﾄｰｸﾝを合成して下流に流す。
ﾌﾟﾛﾀﾞｸｼｮﾝ・ﾉｰﾄﾞ	条件部の成立 （ﾙｰﾙ毎に1個）	本ﾉｰﾄﾞに達したﾄｰｸﾝは条件を成立させるWMEの組であるので、競合集合に登録する。

ネットワークのノードには4種類があり、各ノードでの処理を表1にまとめる。

Ⅲ．従来方式の問題点

(1)知識ベース記述上の問題点

パターンマッチング処理にReteアルゴリズムを採用し、ＯＰＳ５[4]並みの記述力をもつツールでは、膨大な量のWMEからあるパターンの条件を満たすWMEを選び出したり、あるいは複数のパターンを共通に満足するWMEを選別したりする方法としては、基本的にはWMEのデータ部の一致とか比較・範囲指定（イントラノード）、複数WME間の指定要素の一致や比較（インタノード）、あるいはそれらのＡＮＤ／ＯＲ結合ぐらいしか提供されていない。従ってある条件を満足するデータのなかで指定要素の値が最大というのを表現するには、まずある条件を満足するWMEを任意に一つ選び出し、次に同一条件を満足する他の全てのWMEと比較して、それより大きい値を持つものはないと記述せねばならない。すなわちＭＡＲＫＳ－Ⅱの文法に沿って記述すれば、

FACT（優先度　？材料　？優先度）　…… a
～FACT（優先度　？材料　＞？優先度）　…… b
〔～はそれに続くデータが存在しないことを意味する〕

となる。これは利用者の立場からは知識ベースが理解しつらくなり、内部処理的には多大の計算量を要するという問題点がある。いま「優先度」という名のWMEがＮ個あったとする。上記ｂの比較処理は、ａを満足するデータ１個毎に最大Ｎ回ずつ行われる為、結局Ｎの２乗のオーダの比較演算が必要となる。

(2)Reteアルゴリズムに内在する問題点

前節では、最大値や最小値を持つWMEを選出するのにＮ（指定の条件を満たすWMEの総数）の２乗のオーダの計算量を要すると述べたが、Reteアルゴリズムでは部分的なマッチングの状態がインタノードに保存されていることより、実際には新規に追加あるいは削除されたWMEについてのみ、マッチング処理をやり直せば良い。Reteアルゴリズムでは、WMEの任意の要素に設定された固定データによるチェック（データの一致や範囲指定を調べるもので、イントラノードで処理される）を通過したトークンは、インタノードに蓄積された後、パターン間のマッチング処理の対象となる。このときトークンは、初めて流れ込んだインタノードの右メモリに、またトークン間でマッチングが確立した場合はそれらの合成トークンが直後のインタノードの左メモリに格納される。Reteアルゴリズムではこれらのトークンの右／左メモリへの登録順については、何ら言及しておらず、通常は競合解消戦略の関係から到着順に蓄積される。このため新規に追加／削除されたWMEについてのみマッチング処理をやり直せば良いとは言っても、データ群が値の大小順に順序列を構成していないため無駄な処理を多々含むことになる。

以上についてさらに詳しく検討を加える。

＊ＲＵＬＥ　搬送ルール
FACT（搬送指示　？材料　？Ｆ置場　？Ｔ置場　……）
　　…… a
FACT（優先度　？材料　　？優先度）　…… b
～FACT（優先度　？材料　＞？優先度）　…… c
－＞

これはある材料搬送システムにおいて最大の優先度を持つ搬送指示情報を選び出すルールである。いま下図のように搬送指示FACTがM個、優先度FACTがＮ個あるとし、部分的なマッチング状況がインタノードに保存されているものとする。また簡単のために優先度FACTの優先度は全て異なるものとする。

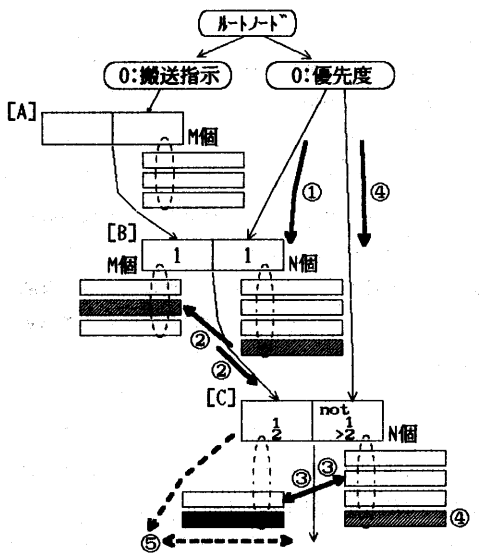


図２．最大値のマッチング処理

ここで優先度FACTが新規に登録された場合を考える。この場合は、＋の符号を持つトークンがルートノードより流され、以下のように処理が進行する。

①ｂのパターンには固定値チェックがないため、FACTのｵﾌﾞｼﾞｪｸﾄ名のチェックの後、［B］のインタノードに達し、まず右メモリに格納される。

②次に反対側の左メモリに格納されているトークンとパターンマッチングを行う。一致するものがあればそれらの合成トークンを作り下流に流す。

③②で作成された合成トークンは次の［C］の左メモリに格納された後、反対側の右メモリと第一要素の一致と第二要素の比較処理を実施する。

④ｂ、ｃについては同一種類のWMEに関する条件であることから、①のｵﾌﾞｼﾞｪｸﾄ名チェックのイントラノードが分岐し、今回新規に作成された優先度FACTは、［B］と共に［C］にも流される。［C］では、［B］と同様まず右メモリにトークンが格納される。

⑤ｃは「存在しない」ことを意味するＮＯＴパターンであることから、この場合は反対側の左メモリを調べ、右メモリ側に一致と比較の条件を満たすFACTが存在しないことによりＮＯＴ条件を満足している合成トークンをサーチする。条件を満たすトークンが存在すれば、今回の新規FACTの追加によりこの条件の成立が侵害されていないかを調べる。条件を満足できなくなった場合には、下流のノードに対して、その旨を知らせるトークンを順次流す。

次に計算量について考察する。オリジナルのReteでは、通常インタノードへの到着順で右／左メモリのトークンの管理を行っている為、①については右メモリの末尾にトークンをつなぐだけで処理が終了する。②については左メモリの全てのトークンとマッチング演算をする必要がある為、M回の演算が行われる。③の処理は②で一致したものについてのみ行われるが、一致したもの１個につき最高Ｎ回のマッチング演算が必要となる。⑤ではＮ回のサーチ演算が行われる。結局、M＋Ｎのオーダの計算量となる。

今度はある優先度FACTが削除された場合を考える。この場合は－の符号を持つトークンがネットワークに流される。

①WMEの削除を意味する－トークンは、FACTのｵﾌﾞｼﾞｪｸﾄ名のチェックの後、［B］のインタノードに達する。今回は－トークンの為、右メモリ内に同一内容のトークンがあるかを調べ、存在する場合はそのトークンを消去すると同時に反対側の左メモリにマッチングしているトークンがないか調べる。ここでは右メモリに関して高々Ｎ回と左メモリに関してM回のマッチング演算が行なわれる。

②①の左メモリにマッチングするものがある場合は、－の符号を持つ合成トークンが［C］に流され、左メモリから同一内容を持つトークンを消すと同時に、それが反対側の右メモリとマッチングしていなければ、さらに下流に対して合成トークンを流す。

④－トークンは、［C］にも流される。［C］の右メモリに関しては、①と同様、同一内容を持つトークンが消去されるが、［C］ノードは、ＮＯＴ処理を行うノードの為、左メモリの処理については壊滅的な事態が発生する。即ち今回の－トークンにより、条件成立の障害となっていたトークンがなくなったかも知れないため、全ての左要素に対して、右要素とのマッチングをやり直す必要がある。いま優先度FACTがＮ個あるとすると、Ｎの２乗のオーダの再計算が必要となる。

Ⅳ．新方式による解決策

（１）最大値・最小値の取り扱い

最大・最小に関する以上の問題点は、Reteアルゴリズムでは部分的なマッチングの状況をインタノードに保存しているものの、大小関係による順序づけた管理を行っていないことに起因していた。そこで本稿では、インタノードの左あるいは右メモリにおけるトークンを、注目する要素の値の昇順あるいは降順で管理することにより、知識ベース記述の簡潔化とマッチング処理の高速化を同時に達成する方式を提案する。

『インタノードの右メモリ、あるいは左メモリごとに、各々以下の情報を持ち、最大値・最小値指定の場合は値の降順あるいは昇順で、例えば２重リンクリストのような柔軟なデータ構造によってトークンを管理する。

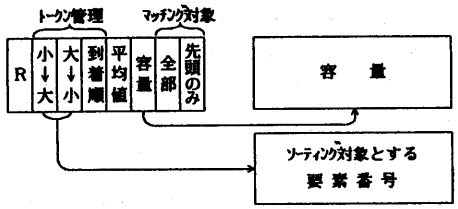


図３．ノードメモリの管理情報

また最大値・最小値指定の場合は、先頭に位置するトークンのみをマッチングの対象とする。但し、２番目以降が先頭と同一値の場合は、２番目以降であってもマッチングの対象とする』

本方式を知識ベース上に記述可能とする為に、！ＭＡＸと！ＭＩＮの２つの記述子を導入する。但し、両者とも数値型の要素に対してのみ、指定可能とする。この記述子を使用すれば、前述の問題は以下のように簡潔に表現でき、可読性も飛躍的に向上する（ノード数の減少にもなる）。

＊ＲＵＬＥ　搬送ルール
FACT（搬送指示　？材料　？Ｆ置場　？Ｔ置場　……）
……a
FACT（優先度　？材料　！ＭＡＸ＆？優先度）　……b
－＞

次に計算量について評価する。前章と同様に、まず優先度FACTが新規に追加された場合を考える。なおｂパターンの右メモリでは！ＭＡＸ指定により、トークンは２番目要素の降順にソーティングされているものとする。

①優先度FACTのトークンは、ｵﾌﾞｼﾞｪｸﾄ名のチェックの後、［Ｂ］のインタノードに達する。今回ノードに到達したトークンの値と、右メモリに格納されているトークンの値を、右メモリの先頭から順に調べ、新規にノードに到達したトークンの格納場所を特定する。新規FACTが最大の優先度を持つ場合は１回で、そうでない場合は平均Ｎ／２回の比較演算で格納場所を特定できる。

②！ＭＡＸ指定のインタノードでは、マッチング処理は先頭のトークンのみを対象とするため、格納位置が２番目以降であれば、マッチング処理は行わない。いま新規の優先度FACTが最大優先度を持つならば、先頭のトークンが入れ替わるので、まず現在先頭に位置するトークンについてマッチングの解消を行った後、新規のトークンにより、左メモリとの間でマッチング処理を実施する。左メモリにはＭ個のトークンが格納されている為、Ｍ回のマッチング演算を行う。

③その後の処理については、従来と全く同じである。

結局、最大値トークンが入替わる場合はＭ回、入替わらない場合は平均Ｎ／２回の演算で済む。さらに今回は、余分なマッチング状況は作り出されない為、自由メモリの大巾な節約にもなるし、左メモリに対する処理も軽減される。

次にートークンが流された場合を考える。

①［Ｂ］のノードにートークンが達した場合、右メモリに同一内容をもつトークンがあるかを調べ、存在すれば消去して消去情報を下流に反映する。右メモリは降順にソーティングされている為、右メモリのトークン数がＮ個であれば平均Ｎ／２回の比較演算で済む。

②ートークンにより最大値が入替わった場合は、次点のトークンによりマッチングをやり直す（Ｍ回）。入替らない場合はマッチング演算は必要ない。

すなわちオリジナルのReteの場合、ートークンに対しては、Ｎの２乗のオーダのマッチング演算を必要としたのに対し、今回は高々Ｍ回の演算で完了する。

（２）極大値・極小値の取り扱い

Ｉ章で例示した知識につき、図４を考えてみる。

いま左側のルール中では、最小重量と最大含有量をそれぞれ！ＭＩＮと！ＭＡＸで表現している。ここでマッチング状況を調べてみると、ａのパターンでは重量FACTにつき第２要素の最小のものに着目してＡが選ばれ、ｂのパターンでは成分FACTの第３要素の最大のものという観点からＢが選ばれている。本ルールでは、ａ，ｂパターンとも第１要素は「？材料」というパターンマッチング変数が使用さ

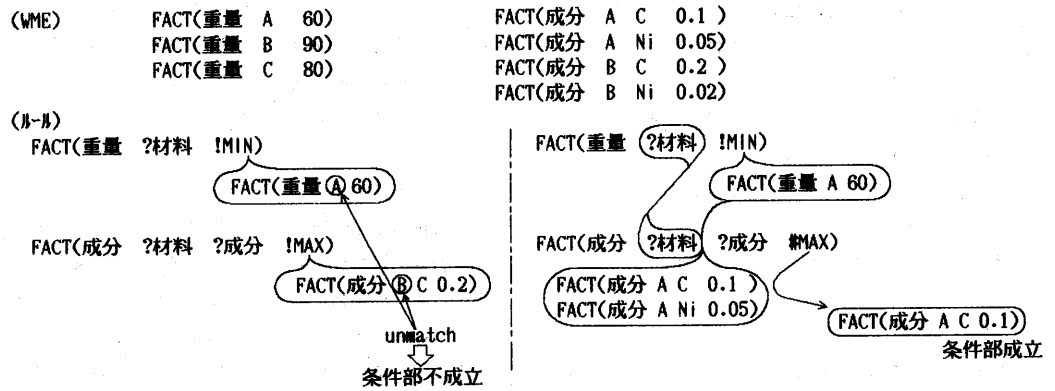


図４．最大・最小の意味の相違

れており、両者が一致する必要があるが、この場合aパターンではA，bパターンではBとバインドされている為、結局条件部は成立しなくなる。これは、前節で示した！ＭＡＸ，！ＭＩＮは、固定データによるチェックを満足した（特定のイントラノードの並びを通過した）WME群の中で指定した要素が最大あるいは最小の値をもつWMEを選出するものであった為である。

そこで新たに#ＭＡＸと#ＭＩＮという２つの記述子を導入する。これは意味的には図４に示すように、指定したパターンまでの間でパターンマッチングをしたWMEの組から、ある要素に着目して最大あるいは最小の値（即ち部分的にマッチングした部分集合内での最大・最小）を持つデータ組を選出するものである。すなわち！はイントラノードを通過したデータを対象としているのに対し、#はインタノードに於けるマッチングが成立したデータを対象としたものである。

（３）Reteネットワーク上でのインプリメント

！については、イントラノードを通過したデータを対象としたものであり、２入力１出力の形をとるインタノードに対しイントラノードからは必ず右方向から装入するので以前に示したように、右メモリを昇順あるいは降順にソーティングして管理することにより容易に実現できた。

これに対し、#はインタノードに於けるマッチングが成立したWMEの組を対象としたものである。ところで、あるインタノードでマッチングが成立した場合は、それらの合成トークンが作られ下流に流される。下流はもちろんインタノード（プロダクションノードも含む）であり、その左方向から装入される。すなわち#の場合は、ルールで記述されたパターンの次パターンに対応するインタノードの左メモリを、指定要素の昇順あるいは降順にソーティングして、管理すれば良いことになる。

以上まとめると、インプリメント上は、

！ＭＡＸ，！ＭＩＮ：

　　指定パターンに対応するインタノードの右メモリ

#ＭＡＸ，#ＭＩＮ：

　　指定パターンの次パターンに対応するインタノードの左メモリ

を、昇順あるいは降順で管理すればよい。

（４）知識ベースの記述

ＭＡＲＫＳ－Ⅱでは、FACT，フレーム，プログラム変数等，全ての知識データオブジェクトに関して、ある要素が最大とか最小とかを表現する場合

　　最大（極大）：　！ＭＡＸ　　#ＭＡＸ

　　最小（極小）：　！ＭＩＮ　　#ＭＩＮ

という記述子を使用することにより簡潔に記述することができる。たとえば

　　FACT（関係　１０　？　！ＭＡＸ　？ｘ）

は、第１要素が１０という値を持つ「関係」という名のFACTのうち、第３要素の値が最大のものを意味する。ここで、！は、各々のルールパターンに関して固定データ指定による条件を満足したオブジェクトの中で指定要素が最大あるいは最小を意味し、#は各ルールの先頭から指定パターンまでの範囲でパターンマッチングしたオブジェクトの組の中で、指定した要素に注目した場合の最大（極大）あるいは最小（極小）を意味する。

但し、

①数値型の要素またはスロットに限る。

②同一パターン内で記述できる個数は、（！ＭＡＸ，！ＭＩＮ）のうちの１個と、（#ＭＡＸ，#ＭＩＮ）のうちの１個、計２個までである。

③ＮＯＴパターン内では、！，#とも使用不可。

④最大値や最小値をバインドして後方で使用したい場合は、従来通り？変数を使用する

　　例）　　！ＭＡＸ＆？ｘ

次に知識ベース記述上の注意として、一見同じ意味をもつように見受けられるが、実は全く別の意味を持つ記述例を挙げる。

［A］　FACT（成分　？x　？y　>0.08&！MIN&？z）

［B］　FACT（成分　？x　？y　！MIN&？z）

　　TEST（？z>0.08）

［A］は３番目の要素が0.08以上のものの中での最小値を意味し、図４のデータの場合は？ｚに0.1がバインドされる。これに対し、［B］は３番目の要素の最小値が0.08より大きいか否かを意味し、この場合は？ｚに0.02がバインドする為、条件は成立しない。

（５）新方式の効果

！単体の定量的評価については既に述べた。実際レベルの総合効果については、パターンマッチング処理が、今回の！や#の出現頻度、ルール数や各ルール毎のパターン数、WMEの数、それにルールの記述方法などに依存し、これらの要素が複雑に絡み合うため、定量的に評価することは極めて困難である。本機能を適用した実システムから、経

験的に得られた定性的な効果について以下に列挙する。

①一般的には同一のｵﾌﾞｼﾞｪｸﾄ名を持つWMEの数が増大するほど効率が低下するが、今回の！や＃を導入すればWME数への依存度が小さくなる。特にWMEが多い大規模システムほど効果が大きい。

②余分なインタノード（ＮＯＴ）がなくなり、かつ無駄な部分マッチングが防止されるため、自由メモリの大幅な節約になる。

③記述力が向上し、知識ベースの見通しが良くなる。自然な表現となる。

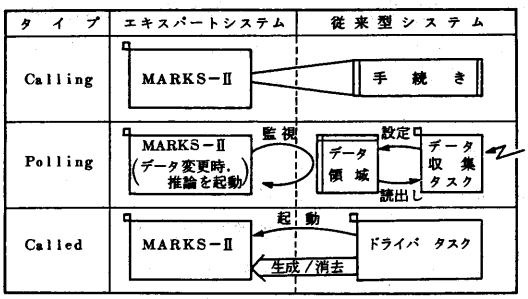
④＃よりも！の方が速度面、メモリ面とも効果が大きい。これは！がインタノードにおけるマッチング前、＃がマッチング後を対象にしていることからも明らかである。

Ｖ．リアルタイム機能への応用

（１）従来型システムとの協調

ＭＡＲＫＳ－Ⅱでは、従来型制御システムと協調した複合型のＥＳを容易に構築できるよう、設計当初から従来型システムとエキスパートシステムとの密接なデータリンケージを非常に重視して開発を進め、現在は表２に示す３種類の型態[5]を提供している。なお、これらは混在が可能で、応用システムのニーズに応じて組合せて使用できる。

表２．従来型システムとの協調型態



Calling型は、ルールの右辺（実行部）で、従来型システムの手続きを直接呼び出せる型のものである。

Polling型は、プロセスデータ等に割り付けられた従来型言語の変数を、ＥＳのルールから直接参照したり更新したりできるもので、ＭＡＲＫＳ－Ⅱの大きな特徴の一つである。表２で、データ収集タスクは、なんらかの外部要因によって割込起動され、プロセス信号入出力装置からデータを取り込み、データ領域にセットしている。推論エンジンはこのタスクとは独立に動作し、周期的にデータ領域をポーリングする。そして外部の変数値と対応する知識データオブジェクトの値に相違があれば、その変更内容を内部に反映し、条件部を満足するルールを実行する。ルールを記述する専門家は、知識ベースの最初の部分でこれらの変数のデータタイプを宣言するだけで良く、外部データを獲得する際の複雑で微妙なタイミングを考慮しなくて済む。

Called型は、従来型システムからＥＳを、手続き呼出しの形で呼出すことができるものであり、以下の３種類の手続きを装備している。

1⁰ 推論の起動

2⁰ 知識データ・オブジェクトの生成（assert）

3⁰ 〃 削除（retract）

この型のリンケージ方法を使えば、従来型システムからＥＳを、必要なときにいつでも起動することができる。

（２）周期データの取扱い

外部データのハンドリングは、Polling型とCalled型の２つの型態で可能である。Polling型はデータ自体は簡素な型だが、データ収集処理が複雑な場合に、またCalled型は比較的多量の情報を含むデータを扱う場合にと使い分けられる。このうちPolling型は、外部データの変化をＭＡＲＫＳが常時監視し、ルールの条件部が満足された時点で推論実行を起動するタイプであるが、運用上次の問題点を含んでいる。

①推論実行中は、Called型による推論起動時、知識ユニット切換え時、および蓄積トークンバッファ[3]満杯時に外部データをポーリングしているが、このタイミングを外部からは把握しにくい。

②推論のアイドリング中は一定周期毎に、外部データをポーリングしており、ＣＰＵ負荷が重い。

そこで今回、Polling型のプログラム入力変数について周期データの取扱いを新設することにする。これは、知識ベース記述のデータ定義の際に個々のプログラム入力変数ごとに独立したサンプリング周期の指定を可能とし、ルール条件部で、ある外部データの

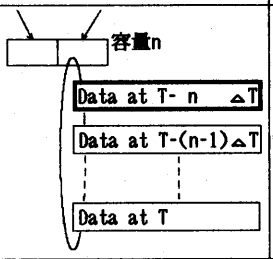
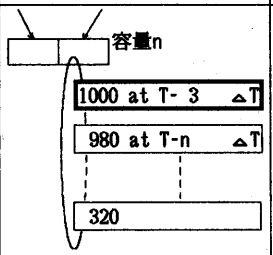
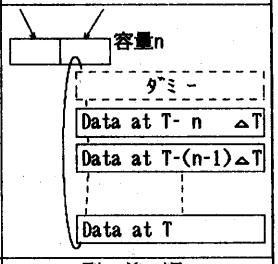
a ｎ周期前のデータ

b 過去ｎ周期間の最大（最小）値

c 過去ｎ周期間の平均値

を扱えるように拡張するものである。推論実行時、推論エンジンは指定されたサンプリング周期毎にプログラム入力変数の値を取込み、タイムスタンプをつけてReteネットに流す。なお周期指定されたプログラム入力変数については、上記①②のタイミングでのデータ監視は行なわない。

表3．Ｒｅｔｅネットワークにおける周期データの取扱い

種　別	n 周 期 前	過去n周期間の最大(最小)値	過去n周期間の平均値
ノード型態			
トークン管理	到　着　順	降順 または 昇順	到　着　順 ＋ ダミートークン(常時先頭)
マッチング対象	先 頭 の み	先 頭 の み	先 頭 の み (再マッチングは合計値の変化時のみ)
トークン受信時の主な処理	•右メモリの容量は(n+1)とし、上流からトークンが流れてくるとメモリ・リンクの末尾にトークンをつなぎ、同時に先頭のトークンをリンクからはずす。	•右メモリの容量は(n+1)とし、上流からトークンが流れてくるとリンク中の適切な場所に格納する。同時に最古のトークンをリンクからはずす。	•メモリ・リンクの先頭に、トークンの合計値を保持する為のダミートークンを設ける。 •容量は(n+1)+1=n+2とする。 •トークンをリンクにつなぐ場合は、ダミートークンの合計値を加算し、はずすときは減算する。

（3）周期データのインプリメント

Reteネットワークに於ける周期データの取扱いは、今回の最大・最小機能を応用すれば、インプリメントは表3に示すように比較的容易に行える。

今回は周期データの取扱いはプログラム入力変数のみとするが、他の型の知識データオブジェクトについても同様に拡張可能である。ところで通常のReteネットワークでは、イントラノードを通過できないトークンは消去されるが、今回の場合は指定の時間範囲をトークンの個数で管理しているため、イントラノードの条件を満足しない場合でもインタノードにダミーのトークンを流す必要がある。ダミートークンは個数の管理にのみ使用するもので、イントラノードではチェックなしに下流に流し、インタノードでは右メモリに保存するもののマッチングの対象とはしない。

VI. 結言

プロダクションシステムの枠組の中で、対象データの最大値とか最小値とかを扱う場合、高速のパターンマッチング手法であるReteアルゴリズムを採用したとしても最大データや最小データの選出にはかなりの計算時間を要していた。そこで本稿では、Reteネットワーク内でWMEを順序列として管理することにより、最大・最小の取扱いを大幅に効率化できる方式を提案し、Reteネットワーク上での実現方法を示すとともにその効果について述べた。さらにリアルタイム機能への応用として、本方式を拡張すれば、サンプリングや時間概念を有するデータについても、Reteネットワーク内で簡潔にかつ高速に処理できることを示した。

本稿で述べた最大・最小機能は、既にインプリメントを完了し、大規模な搬送スケジューリングシステムに適用して、その有効性を確認している。またリアルタイム機能への応用については、現在、詳細仕様の詰めを行っている。

［参考文献］

[1]横井、竹中:「プロセス制御向エキスパートシステム構築ツール(1)－概要－」、情報処理学会第35回全国大会,1987.

[2]Forgy,C.L.:"Rete:A Fast Algorithm for the Many Pattern/Many Object Pattern Matching Problem", Artificial Intelligence 19,1982.

[3]竹中、横井:「プロセス制御向エキスパートシステム構築ツール(2)－高速推論機構－」、情報処理学会第35回全国大会,1987.

[4]Forgy,C.L.:"OPS5 User's Manual", Technical Report CS-81-135,Department of Computer Science,CMU,1981.

[5]K.Takenaka et al.:"Expert System Building Tool for Procss Control:MARKS-II", International Workshop on Artificial Intelligence for Industrial Application,1988.