人　工　知　能 98－6
（1995．1．18）

個体群の共通性に着目した ルールベース・マルチエージェントシステムの高速処理方式

砥崎 賢一　　清水 浩

九州工業大学 情報工学部 電子情報工学科

現実的な仮想空間を実現するためには，多数のエージェントがそれぞれ自律的にしかも同時に行動できる世界を作らなければならない．これは，エージェント群の実時間知識処理が必要であることを意味している．そこで我々は，エージェント群が共通の知識源を共有することに着目し，メモリ効率の良い高速推論方式を提案する．提案方式では，ワーキングメモリ要素の量子化と，それに基づく状態空間の移動により照合と競合解消を一括して行なっている．異なるエージェントであっても同一の知識源が状態空間を共有することで，エージェント当たりのメモリ消費量を低く抑えている．

A Fast Inference Method for Rule-Based Multi-Agent Systems focused on a common characteristics of agents.

Ken'ichi KAKIZAKI and Hiroshi SHIMIZU

Department of Computer Science and Electronics,
Faculty of Computer Science and Systems Engineering,
Kyushu Institute of Technology, Iizuka, 820 Japan

In a realistic virtual environment, a large number of agents must act autonomically at the same time. We propose a fast inference method for real-time rule-based multi-agent systems. In our method, state space concept for knowledge sources is introduced, and quantum values calculated from working memory elements are used to categorize each knowledge sources in a state space. In every inference cycles, matching and conflict resolution are performed enbloc by categorizing knowledge sources in the space. The node of categorized knowledge sources in the space indicates actions to do in the state, so the method realizes fast inference systems.

1 まえがき

今後のマルチメディア社会の到来にともない，仮想現実システムへの期待が高まっている．現実的な仮想空間を実現するためには，仮想空間内に多数の人間や動物を生成して，現実世界と同様に振舞わせることが強く望まれる．このためには，多数のエージェント群の実時間知識処理を行なわなければならない．

エージェントの知識処理を行なうには，現在エキスパートシステムの構築に広く用いられているプロダクションシステム [小林 85] を利用することができる．しかしながら，従来のプロダクションシステムを利用して多数のエージェントの実時間知識処理を単純に実現しようとすると，処理時間やメモリ効率の面で非常にコストが大きくなるという問題がある．

複数エージェントの知識処理を高速化するために，並列・分散処理の研究 [小林 85][横山 93] [金井 92] がなされている．これらの研究では，複数のエージェントがそれぞれ異なった知識によって推論を行なうものが対象となっている．しかしながら，多人数の人間のように同種類のエージェント群について考えてみると，その知識は個人によって異なる部分があるとともに，手足の動かし方，反射的な動きなどの共通部分が多数ある．このようにエージェント同士の共通性に着目することは，知識源を共有できるようになるという点で重要である．

本論文では，このように共通部分を持つエージェントが多数存在することを前提として，複数のエージェントが独立に知識処理を行ない行動するシステムにおける高速な推論方式を提案する．提案方式では，コンパイル時に知識源のプロダクションルールの左辺の解析によって状態空間を作成する．推論サイクルでは，ワーキングメモリの要素値の量子化，状態空間の移動という簡単な操作によって，照合処理と競合解消を一括して高速に行なっている．

2 エージェントモデル

我々の提案方式が仮定しているエージェントのモデルを以下に示す．

エージェントは，その振舞いを性格付けるいくつかの要素によって構成されている．構成要素としては，図 1に示すようにそのエージェントの状況を表すワーキングメモリ，知識源および外部との入出力機能がある．これらが有機的に結合されることによってエージェントが自律的に行動を始める．ワーキングメモリは，知識源の実行部，あるいはエージェントの外界の情報を取得する入力装置によって変更され，ワーキングメモリの内容に基づき推論を行ない，次の行動を決定する．多くの場合，外部変化に応じて実行部が選択されるという点で，本モデルは事象駆動型 (event driven) となっている．

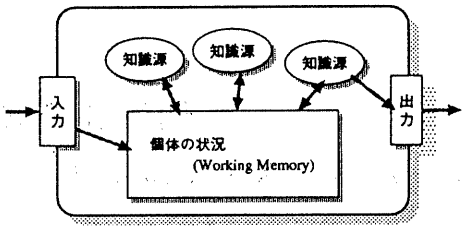

図 1: エージェントのモデル

2.1 知識源の分割

エージェントの振舞いは主に知識源によって規定される．知識源内部では，知識源は対応する問題ごとに分割され定義されている．知識は IF～THEN～の形のプロダクションルールの集合 (PM) で記述する．推論機構は各知識源の PM とワーキングメモリと呼ばれる状況要素の集合を参照・比較しながら推論を進めていく．

エージェントの知識はその処理内容によっていくつかに分類できる．例えば人間であれば，「肉体を動かすための知識源」，「数学の問題を解くための知識源」といったように，知識は問題に対して分担して解決法を求める複数の知識源から成り立っていると考えることができる．

このように知識源を対象ごとに分割すると，そのエージェントの知識を単純な定義の複合体として記述することができ，知識が構造化される．これによって知識の記述性を上げることができる．また，複数の種類の知識源を用意しておくことによって，知識源の組合せによってエージェントの多様性を実現することや，複数のエージェントが一つ

の知識源を共有することも可能になる．

2.2 ワーキングメモリ

ワーキングメモリには，エージェントの内部状態を示す値と，そのエージェントが認識可能なエージェントの外部の状態を示す値が格納されている．内部状態は，プロダクションルールの実行部によって，推論サイクルごとに刻々と書き換えられる．また，外部状態を示す値は，エージェントが持つ入力装置によってエージェントの外部の状態が変化するたびに書き換えられる．

ワーキングメモリは知識源によって参照され，実行されるルールの選択に影響を与える．ワーキングメモリは複数の知識源の通信にも利用されるため，本モデルは黒板モデルに基づくものと考えることができる．ワーキングメモリの参照と変更の様子を図 2に示す．

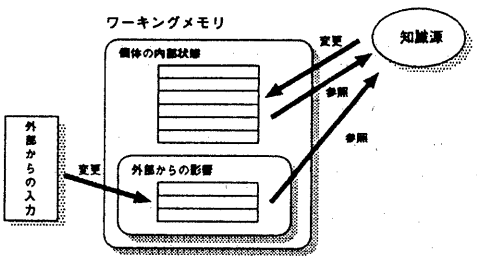

図 2: ワーキングメモリ

2.3 推論

本提案モデルでは，一般のプロダクションシステムの推論実行サイクルと同じように，推論はサイクルごとに繰り返し行なわれる．本論文では，このような一回の推論サイクルをティックと呼ぶことにする．仮想現実システム内のエージェント群をシミュレートする場合，画面の更新間隔と同じ割合で推論を行なおうとすると，すべてのエージェントを構成するすべての知識源に関して，1秒間に10ティック以上推論を行なわなければならない．

本提案モデルでは，一つの知識源の一回の推論で選択され実行されるルールは一つのみである．複雑な問題を解く推論が必要な場合には，ワーキングメモリを書き換えながら数ティックかけて行なう．これは，人間においても簡単なことは反射的に解決し，複雑なことへの判断は時間がかかるという事実と合致している．

2.4 競合解消

知識源とワーキングメモリの照合の結果，選択可能なルールが複数になる場合がある．この場合，実行するルールを一つに特定するために競合解消を行なう．競合解消は，一つの概念を表す知識源内で行なわれ，条件部の制約が最も多いルールを選択するといった，ある一定の競合解消規則によって行なわれる．また，一つのエージェントを構成する各知識源で選択されたルールは，それぞれ別の概念に対する競合しない動作であると仮定し，これらは並列に実行される．

3 処理方式

本提案モデルに基づく多数のエージェント群の推論処理を効率良く高速に行なう推論方式を提案する．本方式では，推論サイクルの実行に先立って，各知識源のコンパイル時にプロダクションルールの解析を行ない，状態空間の生成を行なう．この際，ワーキングメモリの要素値が量子化できるかどうかを調べ，状態空間の大きさを特定するとともに，その増大を抑制する．

プロダクションシステムの実行サイクルは，照合部，競合解消部，実行部の 3 つの処理に分けられる．本処理方式では，実行サイクルの処理時間の多くを占める照合部および競合解消部の処理を一括して行なっているので，推論を高速に行なえるという特徴がある．本方式の処理は大きく次の2つに分けられる．

- ワーキングメモリ値の量子化
- 知識源の状態空間への配置

本方式の処理の流れを図 3に示す．

3.1 ルールの解析による状態空間の生成

ある知識源で選択されるルールの実行部は，その知識源を持つエージェントの状態によって決まる．またエージェントの状態は，ワーキングメモ

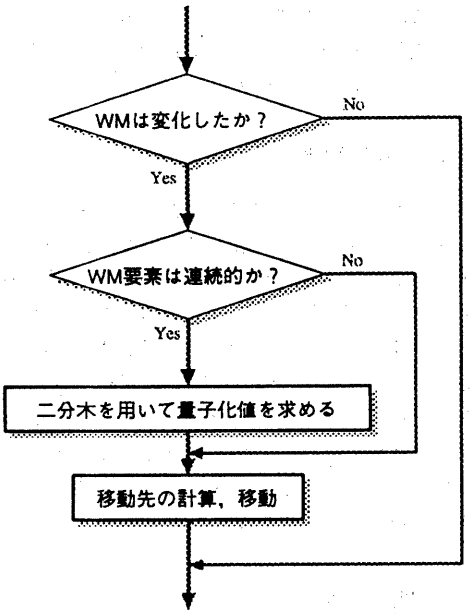

図 3: 状態空間のノード間の移動のフローチャート

リの要素数および各要素の取り得る値によって決まる．従って，知識源に関係するワーキングメモリの要素数とその要素の取り得る値がわかればその知識源の状態空間が定まる．状態空間上の各状態をノードと呼び，ノードにはその状態で選択されるルールを関係付ける．提案方式では，各知識源の照合操作および競合解消は，それぞれが参照するワーキングメモリによって定められる状態空間上のノードに移動させることによって行なう．各知識源には，その知識源が位置するノードを示す指標を持っている．

表 1: かえるのワーキングメモリ

WM 要素	値
動作	鳴く，静か
人との距離	0 ～ ∞

表 2: かえるの鳴く知識源のルール

P1	IF (人との距離<10m) THEN 動作 := 静か
P2	IF (人との距離>20m) THEN 動作 := 鳴く

例えば，表1のようなワーキングメモリと表2の知識源を持ったかえるがいるとする．このかえるは普段はケロケロ鳴いているが，人が 10m 内に近付いて来ると鳴くのをやめ，人が 20m 離れると再び鳴き出す．このかえるの「人との距離」というワーキングメモリの要素は，連続値で無限の状態をとることができる．しかしながらルールの条件部を見ると，「10m より近い」「10m 以上 20m 以下」「20m より遠い」の三つの状態に量子化できる．このように，ワーキングメモリの要素を量子化する際のしきい値はルールの条件部を解析することによって求めることができる．

本方式では，ルールの条件部を解析して量子化のためのしきい値を求め，ワーキングメモリの要素数を n とした場合 n 次元の状態空間を作成する．かえるの例ではワーキングメモリの要素数が 2 個あるので 2 次元の状態空間が作成される．照合処理および競合解消は，知識源をその状態に応じて空間上に位置付けることによって行なう．

3.2 状態変化の検出と状態の量子化

ルールの照合操作は，変化したワーキングメモリの値に影響を受けるもののみを対象に行ない，処理速度を向上させる．変化したワーキングメモリの要素のうち連続値をとるものに関しては量子化を行なう．量子化された値が前ティックと同じであれば，状態空間上の位置は変わらず選択されるルールも変化しないため，量子化される前のワーキングメモリの要素値が変化していても照合処理は行なわない．

ワーキングメモリの要素に連続値をとるものが存在している時，連続値はその状態数が非常に多くなるので，そのままの値で状態空間を作成すると，莫大なメモリを消費する．量子化することによって状態空間のためのメモリ使用量を抑えることができ，さらには高速化にもつながる．数値などの連続値で表現されるワーキングメモリの要素の値を量子化するとき，その値がどの範囲に入っているかをルールの左辺から得られるしきい値との比較で求めなければならない．ワーキングメモリの要素の値を全てのしきい値と比較するコストを下げるために，ワーキングメモリの要素は二分木を用いて量子化を行なう．

3.1節のかえるの例では，「人との距離」という

ワーキングメモリの要素は3つの状態に量子化できるが，あるときの「人との距離」の値を量子化する場合は図 4のような二分木を用いて「人との距離」の量子化値を求める．量子化処理における値の比較は，線形的に行なった場合 n 回ほど必要だが，二分木を用いることで，$\log n$ 回で済み状態数が多くなるほど高速化することができる．

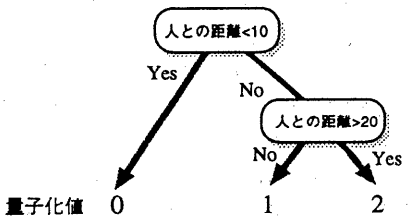

図 4: 量子化値を求めるための二分木

一方，大小関係のない (シンボリックな) ワーキングメモリの要素は，コンパイル時に数値に変換して処理を行なう．この方式では，実行時に量子化の処理を行なわなくてもよく，そのオーバヘッドがない．従って，シンボリックな値の処理はさらに高速に行なうことができる．

3.3 状態空間への配置

ワーキングメモリの値による状態空間は，メモリ上に多次元配列として実現される．各知識源には，状態空間を示す配列の開始位置を示す番地と，その状態空間における各自の位置を示す変移が記録されている．同一の知識源を複数のエージェントが持っているとき，その各エージェントの知識源の状態は同じ状態空間内に配置される．

図 5は，3.1節のかえるの例の状態空間である．「鳴く知識源」は「動作」ワーキングメモリ要素が2個，「人との距離」ワーキングメモリ要素が3個の状態を持っているため $2 \times 3 = 6$ ノードある状態空間を持つ．図 5の丸印一個は一匹のかえるが持っている「鳴く知識源」の状態である．A のかえるのワーキングメモリ要素「人との距離」が「10 より近い」の状態から「10 以上 20 以下」に変化した時は，かえる A の「鳴く知識源」の位置が「人との距離」軸方向に 1 ノード移動する．

各ノードには，図 6に示すように，知識源がその状態の時に起動されるルールを記録した配列の

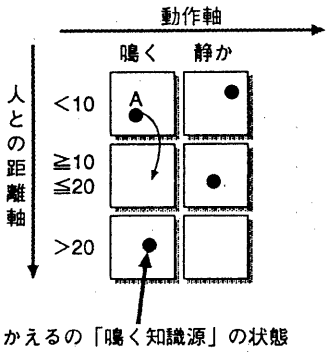

図 5: 状態空間

指標が記されている．ノードに記されている指標の 0 は，選択されるルールがないことを意味している．

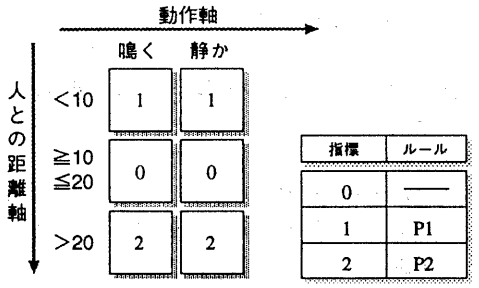

図 6: ノードに記されるルールの指標

図 6の例では，ルール P1 で既にワーキングメモリ「動作」の状態が「静か」であった場合，結果的にワーキングメモリの状態は変化しないのでルールの実行は全く無駄である．このような無駄は，コンパイル時にルールの解析を行なう時に検出できる．この場合には，ノードの指標を 0 に設定することにより，無駄な実行部の処理を除去しさらに高速化することができる．このように最適化を行なった結果，かえるの状態空間におけるノードに結びつけられるルールの指標は，図 7のようになる．

3.4 競合解消

ある状態を満たすルールが複数ある場合，競合解消が必要となる．条件部の制約の強いものを選択するなどの静的な競合解消規則の場合には，競

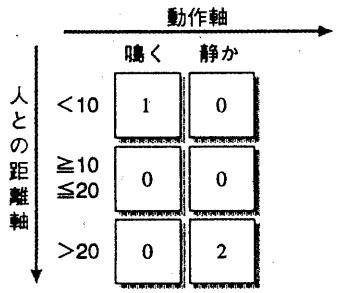

図 7: ノードに書かれているルールの指標の最適化

合解消の結果選択されるルールは，状態空間の作成時に解る．従って，ノードが指すルールを一個に限定して記録できるため，知識源の状態空間中の移動によって照合と競合解消が一括して行なわれたことになる．このため，プロダクションシステムにおける照合処理と競合解消を高速に処理することができる．

図 8の例では,「人との距離>20」で「動作==静か」であるノードで，1 と 2 のルールが競合している．この場合，コンパイル時にあらかじめ決められた競合解消規則によって，そのノードに関係付けるルールを決定する．図 8では，or より and の方が強いという規則でルールが決定されている．

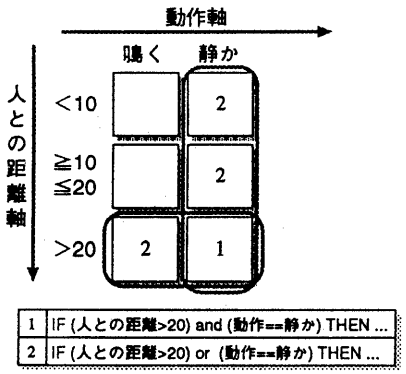

図 8: 競合解消

状態空間を表す多次元配列は，知識源ごとに複数用意することができる．ある知識源に関して，異なる競合解消規則に対応した複数の状態空間を示す配列を作成すれば，各知識源の状態空間を示す配列の開始位置を示す番値を変更することによって，知識源ごとに動的に競合解消規則を変更することができる．また，同じ種類の知識源であっても，異なる状態空間を示す配列を利用することにより，異なる競合解消戦略を利用することができる．

3.5 ノードの移動

ワーキングメモリの量子化値が変化した時には，状態空間を示すメモリ上にマッピングされているそのエージェントの知識源を移動する処理を行なう．知識源の移動は，その要素の量子化値の変更分の値だけ，状態空間のそのワーキングメモリ要素が表している軸に向かって移動すればよい．状態空間は配列で実現されているため，簡単な演算によって移動することができる．

例えば，ワーキングメモリに a, b, c, d という 4 つの要素がある場合，a, b, c, d の四次元配列は一次元配列上に展開されメモリ上に配置される．ここで a が 2 個，b が 6 個，c が 3 個，d が 5 個の量子化値をもつとする．あるとき b の量子化値が 2 から 5 に変化した場合，状態空間の現在の位置から $\{(5-2) \times 15\}$ 番地ほど移動すればよい．ここで 15 は b 以下の c, d の要素の量子化状態数の積で，知識源のコンパイル時にあらかじめ計算しておく．ここで特筆すべきことは一つのワーキングメモリの要素が変更した時に，知識源の状態の移動のための計算は乗算一回で済むことで，本方式の高速化に大きく貢献している．このように状態変化は単純で低コストなノード間の移動で行なわれるので，照合および競合解消の実行を大幅に高速化することができる．

4 評価

4.1 Rete アルゴリズムとの比較

本方式をプロダクションシステムの代表的な高速照合アルゴリズムである Rete アルゴリズムと比較する．

Rete アルゴリズム [For82][石田 88][増位 91] では，ルールの条件部は Rete ネットワークと呼ばれる，内部状態メモリを持つネットワークに変換される．ルールの実行部でワーキングメモリエレメント (WME) が変更されると，それをトークンと

してReteネットワークに流し，ルールの条件部との比較や条件間の変数の一致などの処理を通過したトークンはネットワーク内のメモリに蓄えられる．末端のターミナルノードにトークンが到達すると，そのノードに対応したルールの実行部が実行可能になる．Reteネットワークに変更された全てのWMEを流し終えると，競合解消処理を行なう．本方式ではReteアルゴリズムのこの一連の作業を，状態空間のノード間を知識源の状態が移動するという作業に置き換えたと考えてよい．

まず，Reteネットワークに流れるトークンであるWMEと，本方式のワーキングメモリの要素とを比較する．Reteネットワークに流されるWMEは，

(要素名 ^属性名 属性値 ^属性名 属性値 …)

という，複数の属性の集合で表される．したがって，そのうちのただ一つの属性が変更されただけでも，このWMEはReteネットワークに流されるため，他の変更されていない属性についても照合を行なうオーバヘッドがある．一方，本方式のワーキングメモリの要素は，属性をひとまとまりにせずに単独で扱うので比較に際してReteアルゴリズムで起こったようなオーバヘッドがないという利点がある．

次に，ワーキングメモリの要素と条件部の照合と，ルールの選択について比較する．Reteアルゴリズムでは，変更されたWMEが比較あるいは他のWMEとの変数の一致といった作業を行なうと，競合集合を書き換える操作が必要になる．そしてさらに，その競合集合の中から実行される一つのルールを選択する処理を行なわなければならない．一方，本方式では状態空間の移動という簡単な方法で処理が行なわれ，その知識源の状態の状態空間における位置が決定すれば，それはそのままルールが選択されたことになるので競合解消のコストがいらないという利点がある．

4.2 処理速度

本方式では，現在の計算機が多量に持つメモリ資源を積極的に利用して，プロダクションシステムの処理速度の向上を図っている．本方式が高速なのは照合作業を状態空間のノード間の移動という単純な作業に置き換えているからである．状態空間のノード間の移動操作を示した図3を見ると，条件判断部分で不要な処理を省略しており，(1) ワーキングメモリの変化が少ない場合，(2) 連続値をとらないワーキングメモリ要素の割合が多い場合，(3) 量子化されたワーキングメモリの値の変化が少ない場合に不要な処理を削減でき高速処理が行なえる．

100MHz で稼働する RISC マシンを例として，提案方式が処理できるエージェント数の概算を示す．ワーキングメモリの要素の変化フラグを調べるのに5クロック，量子化に $3+5\times\log n$(n は状態数) クロック，量子化値が変更されたかどうかを調べるのに5クロック，移動先の計算に10クロック，移動に6クロック，その他のオーバヘッドに6クロックかかったとする．このとき，ある知識源の一つのワーキングメモリ要素に関する照合処理は，量子化のしきい値が8個あったとして，50クロックとなる．また，一個体の持つ知識源を平均10個，一知識源が参照するワーキングメモリの変化個数を1ティック当たり平均2個とする．このような条件で一個体当たり10ティック毎秒の推論を行なわせた場合，10,000個の個体の知識源の照合と競合解消を同時に行なうことができると考えられる．

4.3 メモリ効率

本方式で用いられている状態空間をメモリ上に展開する方式は，Reteアルゴリズムの照合･競合解消の一連の処理と比較して高速に処理できる反面メモリ使用量が多くなると考えられる．仮に，エージェントの全ての状態をメモリ上に実現するのであれば，多大なメモリ空間を必要とする．しかしながら，現実的には知識源の分割により，メモリ使用量の増大を抑えることができると考えられる．

ある知識源によって使用されるワーキングメモリの要素数がn個，それぞれの要素がとり得る量子化値の数をそれぞれ $m_1, m_2, \cdots, m_n$，ルールの数を255個以下とすると，ノードに記述されるルールの指標は1バイトで表現できるので，状態空間で使用するメモリ容量は，$(m_1 \times m_2 \times \cdots \times m_n)$ バイトとなる．

例えば，ワーキングメモリの要素数を5個，それぞれの要素がとり得る量子化値の数がすべて4

個の場合は，1kバイトのメモリ空間を占有するだけである．さらに，同じ知識源を持ったエージェントは複数あると考えられ，それぞれのエージェントのその知識源の状態は同じ状態空間を利用する．したがって，仮にそのエージェント数を100エージェントとすると，1エージェント当たりのメモリ使用量はわずか10バイトとなる．このように本方式では，同じ知識源を持ったエージェントの数が増えれば増えるほど，メモリ効率がよくなるという特徴があり，多数のエージェント群をシミュレートするのに適した方式となっている．

5 むすび

現実的な仮想空間を実現するために必要な，エージェント群の実時間知識処理システムとその効率的かつ高速な処理方法を提案した．また従来のプロダクションシステムの高速アルゴリズムとの比較により，本方式が処理速度の面で有効であることを述べた．

本方式では，照合処理に必要な記憶容量が大きいが，エージェント数の増加にともない1エージェント当たりのメモリ使用量を減らすことができる点で有効性がある．また，ルールの条件部の解析や知識源の状態空間の作成など推論の前処理によって1サイクル当たりの推論時間を縮めている．なお，本方式を単純なプロダクションシステムに用いた場合，メモリ消費量の問題は生じるものの，実時間処理に適した高速な推論システムを実現することができると考えられる．

今後の課題としては，本提案方式を仮想現実システムに実際に組み込み，その有用性を実証することが挙げられる．

参考文献

[For82] Forgy, C. L.: "RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", *Artif Intel*, Vol. 19, No. 1, pp. 17–37 (1982).

[横山 93] 横山孝典, 小野昌之, 和田正寛, 大崎宏：“制約とマルチコンテクストに基づく並列強調問題解決”, 情報処理学会論文誌, Vol. 34, No. 2, pp. 342–351 (1993).

[金井 92] 金井達徳, 柴山潔：“分散型プロダクションシステムと並列実行方式”, 電子情報通信学会論文誌 D-1, Vol. J75-D-1, No. 8, pp. 685–693 (1992).

[小林 85] 小林重信：“プロダクションシステム”, 情報処理, Vol. 26, No. 12, pp. 1487–1496 (1985).

[石田 88] 石田亨, 桑原和広：“プロダクションシステムの高速化技術”, 情報処理, Vol. 29, No. 5, pp. 467–477 (1988).

[増位 91] 増位庄一, 田野俊一：“プロダクションシステムの高速化”, 人工知能学会誌, Vol. 6, No. 1, pp. 38–45 (1991).