人　工　知　能 79－7
（1991. 11. 26）

プロダクションシステムを高速化する競合解消方法

市瀬　浩
東洋通信機株式会社

プロダクションシステムにおけるマッチング処理を高速化するためには、マッチング処理の最適化、並列化の他に、マッチングを途中で打ち切る方法がある。
ここでは、マッチングと競合解消を結合した形で行い、競合解消戦略に応じて、必要なインスタンシエーションが得られたか、または、これ以上はインスタンシエーションが生成されないことがわかった時点でマッチングを中止する部分競合解消方式を提案する。
本方式では、値に関する条件を満たしたデータを整列格納しておくことにより、マッチング処理を削減している。

A Process of Conflict Resolution for Speeding up Production Systems

Hiroshi Ichise

Signal Processing Research Laboratory
Toyo Communication Equipment Co.,Ltd.

2-1-1 KOYATO,SAMUKAWA-MACHI,KOUZA-GUN,KANAGAWA 253-01,JAPAN

Among techniques for speeding up the matching process of production systems are optimization and/or parallel processing, and halfway termination of the process.
This paper proposes a process of conflict resolution that executes the matching and conflict resolution in combination, and terminates the matching at a point where it is clear that no more instantiations would be found.
In the process proposed the matching cost is reduced by sort-storing the data that satisfy the conditions with respect to the values.

1。まえがき

実用的なエキスパートシステムを構築するためには、より高速な推論機能が要求され、検討されている［1］。

専門家の知識をプロダクションルールで記述することによって構成されたエキスパートシステムを特にプロダクションシステムと呼ぶ。プロダクションルールは、推論に用いるデータの中から特定のパターンを見つけ出すための条件と、そのパターンを見つけ出したときに行うべき処理を対応づける記述である。プロダクションシステムは、①データと条件の照合判定（マッチング）、②条件をすべて満たしたルールとデータの組（インスタンシエーション）の中からひとつを選択する競合解消、③選択されたインスタンシエーションのルールに記述された処理の実行（ルールの発火）を繰り返し、インスタンシエーションが存在しなくなるか、または，ルールに記述された処理の中で明示的に終了を指定されたされたときに終了する。

ルールが発火すると、一般的にデータの生成・消滅・変更がなされ、これに伴って、インスタンシエーションの生成・消滅が起きる。プロダクションシステムでは，このような状態の変化にともなうマッチングが処理の大部分を占め［2］、マッチング処理の高速化が全体の処理の高速化をもたらす。

マッチングの高速化技術として提案されたＲＥＴＥアルゴリズム［3］では、ルールの発火によってデータが生成・削除・変更された時、そのデータに関する条件の照合判定のみで済むように、ルールの発火以前の条件の充足状態を保存しておき、マッチング処理を軽減している。ところが、データの生成・削除・変更が多く、また、それに関する条件の照合判定が多い場合には、ＲＥＴＥアルゴリズムではかえってマッチング処理が多くなる。そこで、これを改善するため、個々のデータの値についての条件の充足状態を保存し、データの組合せに関する条件の充足状態は保存しないＴＲＥＡＴアルゴリズム［4］が提案されている。しかし、これらアルゴリズムのどちらが適するかは問題に依存するため、一般的な優劣はつけがたい［5］。

また一方では、マッチングを並列処理によって高速化する技術が検討されている［6］。マッチング処理を、個々の照合処理をノードとするデータフローグラフに展開し、これらノードの処理を並列化する場合を、ノード並列という。各ノードの処理を、さらに並列化する場合を、細粒度並列化またはノード内並列という。また、各ルールが独立に記述されていることに着目し、マッチング処理をルール単位に並列化する場合を、ルール並列またはプロダクション並列という。これらの並列化技術は、マッチング処理を展開したデータフローグラフの最適化をしない場合、併用が可能である。

ところで、以上で述べたマッチング方法では、ルールの発火に伴う一連のデータの生成・消滅・変更にともなうインスタンシエーションの生成・消滅をすべて計算し、新たな状態で存在するインスタンシエーションをすべて求めていた（すなわち、競合解消の対象となるインスタンシエーションから成る集合－－競合集合－－の要素をすべて求めていた）。しかし、競合解消を最後まで行った時に求められる筈のインスタンシエーションがその途中で得られたならば、マッチングをそれ以上継続する必要はない。このような考え方に基づき、生成・消滅・変更のあったデータの種類に応じて、最後に求められる筈のインスタ

ンシェーションをいち早く見つけ出すサーチパスをたどることによる、プロダクションシステムの高速化技術が提案されている［7］。

ここでは、競合解消をルール単位の情報に基づいて処理できる部分と全体を通して行う部分とに分割した構成とし、ルール対応の部分をマッチング処理を兼ねて行い、競合解消戦略に依存した形で上記の処理を途中で打ち切ることのできる競合解消方法を提案する。そして、この方法に基づく、プロダクションシステムの構成と並列化について検討する。この方法によって生成され、全体の競合解消で用いられる競合集合は本来の競合集合の部分集合をなすものであるが、以下では、これもまた競合集合と呼ぶこととする。

2．部分競合解消

代表的な競合解消戦略にＬＥＸとＭＥＡがある。これらは、より新しいデータを持つインスタンシェーション、より複雑な条件が記述されたルールを持つインスタンシェーションを優先して選択し、同一のインスタンシェーションの再選択を禁止（リフラクション）する。

ところで、マッチングは、個々のデータの値についての条件照合と、データの組合せに関する条件照合とに分解される。そこで、データの組合せに関する条件照合を、競合解消戦略で規定するデータの新しさの順序で行うことによって、競合解消で選択されないインスタンシェーションの生成を抑えることができる。このように、マッチングと同時に競合解消の一部を行うため、これを部分競合解消と呼ぶ。

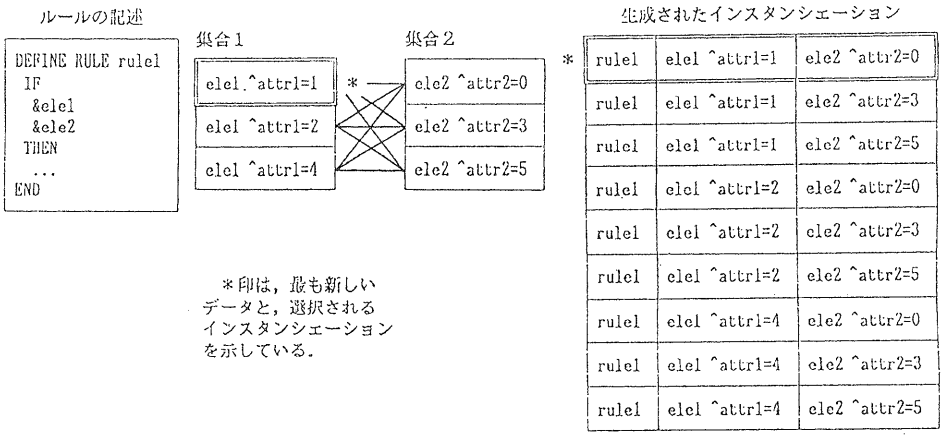
われわれの提案する部分競合解消は、インスタンシェーション生成の中途打ち切りと組合せ生成の早期終了によってプロダクションシステムの高速化を図るものであって、以下にこの２つの視点に立って実現方法を説明する。ここでは、最初の条件にマッチしたデータがより新しい場合を優先するという競合解消戦略を、例として取り上げている。

2．1．インスタンシェーション生成の削減

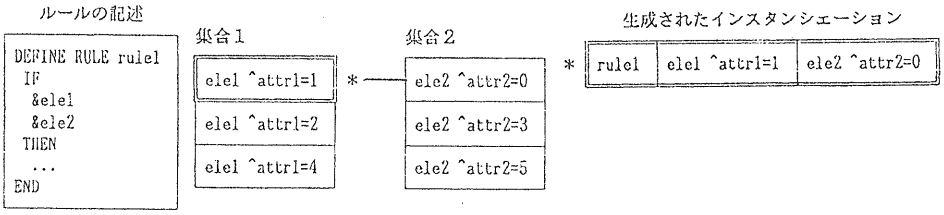
まず、データが生成されたとき、各ルールの各条件に関して、値についての条件が成立したならば、各条件に対応させて、そのデータを新しさの順序で整列させて集合（αメモリ）に格納する。その後、競合解消が要求されたときに、各集合から整列された順序でデータを取り出し、データの組合せに関する条件照合を行う。この条件照合において最初に条件が成立したとき、データの組合せからインスタンシェーションを生成するとともに、組合せに関する条件照合を中止する。

図１は競合集合を完全に生成する場合と部分競合解消をする場合とを比較している。図１（ａ）では、インスタンシェーションが９個生成されているが、（ｂ）では１つに制限されている。このように、部分競合解消では、インスタンシェーション生成処理の中途打ち切りが可能である。

部分競合解消によってルールごとに高々１つ生成されたインスタンシェーションを、そのルールのベストインスタンシェーションと呼ぶ。



（a）全てのインスタンシエーションを生成する場合



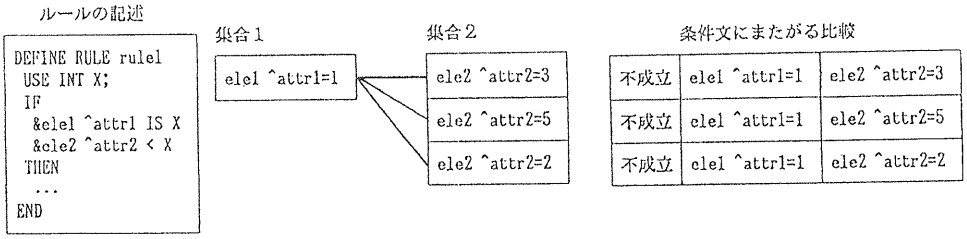
（b）インスタンシエーションの生成を中断する場合

図1．インスタンシエーション生成の中途打ち切りによる高速化

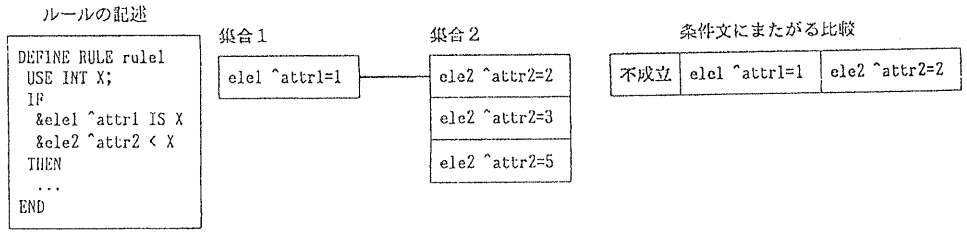
2．2．不要な条件照合の削減

値に関する条件が成立したデータを、のちに行われる組合せに関する条件照合で使用される項目をキーとして整列させ、集合に格納する。その後、競合解消が要求されたとき、各集合から整列順でデータを取り出し、組合せに関する条件照合を行う。このとき、条件の種類（等しい、大きいなど）に応じて、これ以上条件に照合するデータがないことがわかった場合、（ルールごとに）条件照合を終了する。

図2（a）では、インスタンシエーションが1つも生成されないにもかかわらず、集合2からデータを取り出して条件照合を最後まで行わなければならないのに比べて、（b）では、集合2の中のデータには、それ以上^attrの値が小さいものが存在しないことが整列格納によって保証されているため、インスタンシエーションが生成されないことがこの時点でわかり、1回の条件照合だけで処理を終了している。



（a）データが整列されていない場合



（b）データを整列させた場合

図２．インスタンシエーションが生成されない場合の高速化

３．部分競合解消を用いたプロダクションシステムの構成

図３に、一般的なプロダクションシステムの制御構造を示す。すでに述べたように、マッチング、競合解消、ルールの発火を繰り返すように構成されている。

部分競合解消を用いた場合、図４のような構成に変更される。部分競合解消はマッチングの一部と競合解消の一部をまとめて行うため、マッチングはプリ・マッチングと部分競合解消の２段階に分割され、競合解消の残りを行うための全体競合解消が追加されている。

以下に、図４における構成要素の働きを述べる。

３．１．プリ・マッチング

データの値に関する条件判定を行う。条件が成立した場合、競合解消戦略と、のちの部分競合解消におけるデータの組合せに関する条件に基づいて、データを整列させて格納する。

３．２．部分競合解消

すでに述べたように、プリ・マッチングによって格納されたデータを、競合解消戦略に基づく整列順序で取り出し、データの組合せに関する条件判定を行う。各ルールについて、高々１つのベストインスタンシエーションを生成し、それらによって競合集合を生成する。

３．３．全体競合解消

部分競合解消によって生成された競合集合から、競合解消戦略に基づき、高々１つのインスタンシエーションを選択する。

ここで適用すべき競合解消戦略は、データの新しさについてさらにインスタンシエーションが競合した場合に、条件の複雑なインスタンシエーションを優先することである。

３．４．ルールの発火

全体競合解消で選択されたインスタンシエーションのルールに記述された処理を、そのデータを用いて実行する。

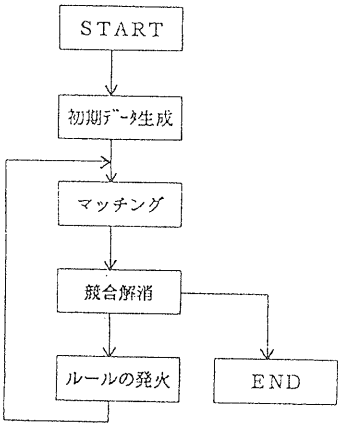


図３．一般的なプロダクションシステムの制御構造

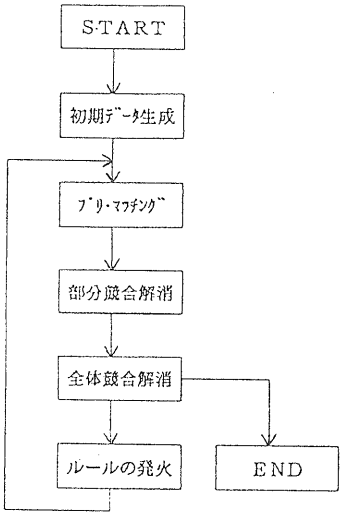


図４．部分競合解消に基づくプロダクションシステムの制御構造

４．並列化

プロダクションシステムにおけるひとつの特徴はルール並列処理化が容易なことであり、また、われわれの提案する部分競合解消においても処理をルールごとに行うので、ここでは、図５に示す構成を提案する。ルール並列化されたプロダクションシステムの基本的な構成方法は文献［８］に従う。本方式ではローカル競合集合が明確な形で存在しない点が異なる。

図４において、並列化可能な要素は、プリ・マッチングと部分競合解消であった。ところで、競合解消戦略におけるリフラクションは、部分競合解消においても適用されなければならない。したがって、図５に示すように選択確定の導入が必要となる。

以下では、図５において図４から拡張された構成要素の働きを述べる。

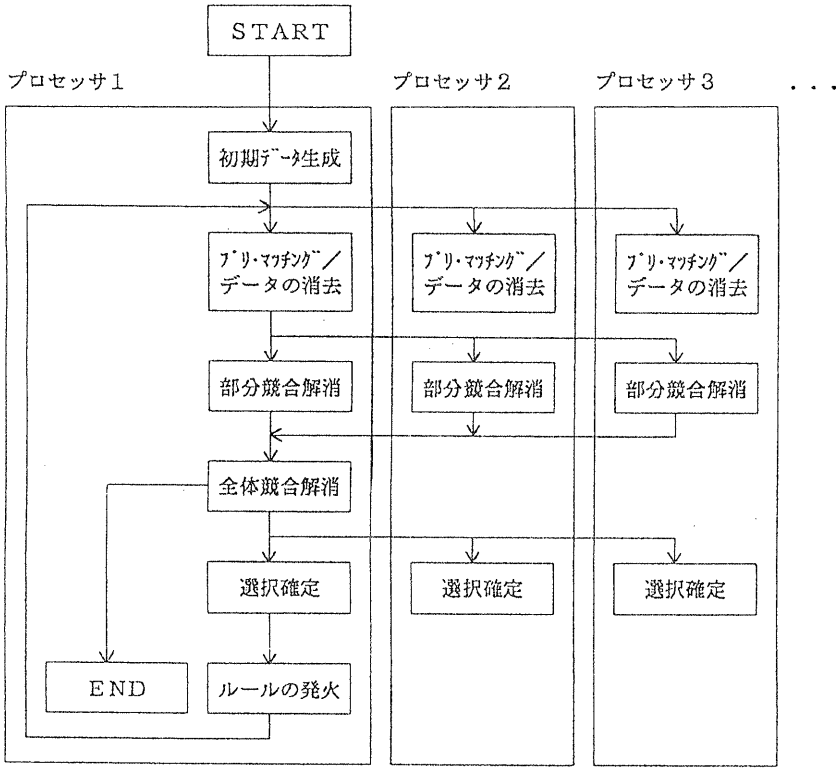


図5．並列化されたときの制御構造

4．1．部分競合解消

並列処理を行う場合は、ベストインスタンシエーションが生成されないならば、その代わりに終了情報を全体競合解消に転送する。それ以外は図4の場合と同様である。

4．2．全体競合解消

部分競合解消を並列処理する場合、すべての部分競合解消処理が終了してベストインスタンシエーションまたは終了情報がそろうまで待つことを除いて、図4の場合と同様である。

4．3．選択確定

リフラクションを実現するため、全体競合解消が選択したインスタンシエーションを履歴情報として記憶する。

５．むすび

プロダクションシステムにおける、競合解消戦略に依存した形の高速化手法である部分競合解消と、これに基づく並列処理について述べた。

各ルールに関して１度に高々１つのインスタンシエーションを生成することと、データを競合解消戦略にしたがって整列しておくことにより、インスタンシエーションの生成処理を短縮できる。

現在、ＭＥＡに類似の競合解消戦略を用いたプロダクションシステムを生成する言語処理系を作成し、これを用いてのプロダクションシステムの試作中である。

［参考文献］

［1］石田，桑原：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)

［2］石田　亨：プロダクションシステムと並列処理：情報処理，Vol.26，No.3，pp.213-225(1985)

［3］Forgy,C.L.：A Fast Algorithm for Many Pattern/Many Object Pattern Match Problem：Artificial Intelligence，Vol.19，No.1，pp.17-37(1982)

［4］Miranker,D.P.：TREAT:A Better Match Algorithm for AI Production Systems：Proc. of AAAI-87，pp.42-47(1987)

［5］荒屋眞二：状態保存式パターン照合アルゴリズムの定性的コスト分析と最良アルゴリズム選択法：人口知能学会誌，Vol.6，No.3，pp.436-439(1991)

［6］Gupta A.：Parallelism in Production systems：Pitman / Morgan Kaufman (1987)

［7］Miranker,D.P.：On the Performance of Lazy Matching in Production Systems：Proc. of AAAI-90，pp.685-692(1990)

［8］南山智之：プロダクション・システム記述言語ＳＩＬＫについて：第３４回情報処理全国大会，pp.1619-1620(1987)