1H−5

μPSにおける プロダクションシステム高速化技法

市瀬　浩　　宮崎　義実
東洋通信機株式会社

1　はじめに

プロダクションシステムにおける認知−実行サイクルは，ルールの発火，競合集合の生成および競合解消の繰り返しである．この中で，競合集合を生成するには膨大なデータの組合せの中からルールの条件を満たすものを選別しなければならないので，その処理の高速化が検討されている．

我々が開発しているプロダクションシステム構築用言語処理系μPSでは，μPSコンパイラが生成する出力コードの中で，競合解消で選ばれない不要なインスタンシェーションの生成を抑えて競合集合の生成から競合解消までの処理を高速化する方式（部分競合解消方式）を採用している．

本稿では，ひとつの競合解消戦略を設定して部分競合解消処理の手続きを作成し，部分競合解消方式を適用した場合と適用しない場合とでの実行時間を比較した実験結果について述べる．

2　部分競合解消処理手続きの作成

部分競合解消［市瀬 91］でのマッチング木は図1のようになる．

部分競合解消方式では，このマッチング木にしたがって，推論サイクル毎に各ルールから高々1つのインスタンシェーションを生成したうえで全てのルールにわたる競合解消を行っている．各ルールから生成される1つのインスタンシェーションは，他のルールがなかった場合には競合解消で選ばれるものである．そして，このようなインスタンシェーションを生成しやすいように，また，インスタンシェーションが生成できない場合には早々に処理を打ち切ることができるように，αメモリの中のデータを整列している．

したがって，部分競合解消処理手続きを実現するには，競合解消戦略が既知であることと，αメモリの中のデータの整列方法，αメモリからのデータの取り出し方，および，競合解消戦略を満足するための付加的処理を定めることが必要である．

2．1　競合解消戦略

ここでは，

［戦略1］同一のインスタンシェーションを2度選択しない．

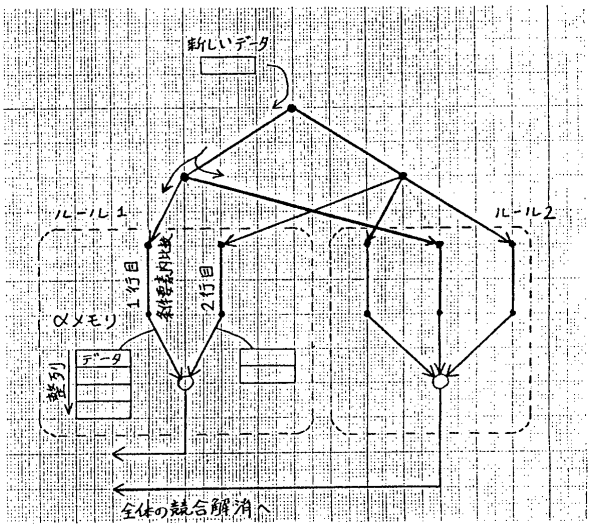

図1：部分競合解消におけるマッチング木

［戦略2］ルール中の1行目の条件にマッチしたデータは新しいデータを優先する．

という競合解消戦略を用いる．

2．2　αメモリの中のデータの整列

データの新しさを識別するため，データの生成時刻をデータに付加する．［戦略2］に基づいて新しいデータを優先させるときには，この時刻の値の大きい順にαメモリの中のデータを整列させる．

次に，ルールの条件要素には，一般的に複数の要素間条件が設定される．そこで，要素間条件があるときには，条件要素の記述順で最も最初に出現する条件で用いている項目をデータの整列キーとして用いる．整列順序は，aをデータのキー項目の値，bを条件における比較対象の値としたとき，比較式がa＞b，a≧bおよびa＝bの場合は大きい順に，a＜bおよびa≦bの場合は小さい順にする．

2．3　データの取り出し方

1回前のサイクルでのルールの発火でデータの生成が行われなければ，αメモリからのデータの取り出し方はその整列順にしたがう．ルールについて一組のデータを

A Method for Speeding-up Production Systems by μPS
Hiroshi Ichise, Yoshimi Miyazaki
TOYOCOM

取り出したならば，そのルールの条件の要素間比較を行い，条件が成立するか条件を成立させるデータが存在しなくなるまでデータの取り出しと比較を繰り返す．αメモリの中のデータを整列させてあるので，あるデータが整列キーの項目による比較で条件成立しなかったならば，同じαメモリの中にはこれ以上条件を満たすデータがないことがわかる．

同じデータの組合せを何回も取り出すことを避けるために，1回前のサイクルで取り出したデータのαメモリの中での位置を記憶しておき，競合解消で選択されてルールの発火で使用された場合は次の位置のデータを，そうでなければ同じデータを取り出す．直前のルールの発火で新しいデータが生成されたならば，［戦略2］の要請を満たすために，次に取り出すデータの位置の記憶を放棄して先頭のデータを次に取り出すようにする．

2．4　付加的処理

部分競合解消方式では1回のサイクルで各ルールから高々1つのインスタンシェーションしか生成しないので，競合集合は小さいが完全ではない．したがって，［戦略1］の要請を満たすために，競合解消で選択されたことのあるインスタンシェーションを各ルール毎に記憶しておく必要がある．データの取り出し位置の変更後のサイクルではこの記憶を参照しながら，競合解消で選ばれたことのあるインスタンシェーションを避けてインスタンシェーションの生成を行う．この記憶の中にあるインスタンシェーションがデータの削除などによって存在できなくなった場合は，この記憶の中から消去する．

3　実験結果

簡単な例として，

人間の集団を，一定の人数の組にわける．
ただし，同一の組に属する人間の年齢はすべて異なる．

という問題を用いて，部分競合解消方式の適用による処理速度の改善の度合を求めた．

この問題をμＰＳ言語で記述すると，ルールは図2のようになる．

```
DEFINE RULE Grouping
 USE INTEGER X1,X2, STRING N1,N2,N3;
 IF
   who*    ^age IS X1              ^name IS N1
  &who**   ^age # X1  ^age IS X2   ^name IS N2
  &who***  ^age # X1  ^age # X2    ^name IS N3
 THEN
  MAKE  group ^one N1  ^two N2  ^three N3;
  REMOVE  who*, who**, who***;
END {End of definition}
```

図2：組わけルールのμＰＳ言語による記述

図中，記号 # は「等しくない」ことを表している．一組の人数を増やせば，IF から THEN の間にある条件要素の行などは増える．

この例を用いて，部分競合解消方式を適用した場合と適用しなかった場合の実行速度の比を図3に示す．実行速度は部分競合解消の適用によって数倍から10倍に向上し，ひと組の人数が多くなるにしたがって実行速度の比は大きくなった．ひと組の人数と全体の人数を増やして行くと，さらに大きな実行速度比が期待できる．

次に，複雑な例として，製造工程のスケジューリング問題についての実験結果を図4に示す．この場合でも，実行速度は30％向上した．

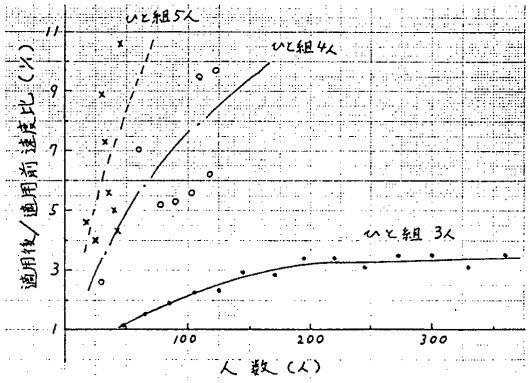

図3：部分競合解消適用前後の速度比-組わけ問題-

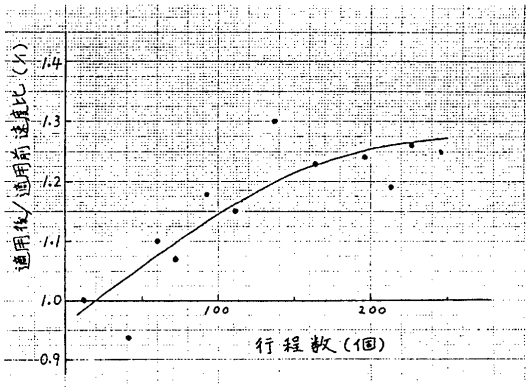

図4：部分競合解消適用前後の速度比-ｽｹｼﾞｭｰﾘﾝｸﾞ問題-

4　おわりに

競合解消戦略に基づいてαメモリの中のデータの整列方法とαメモリからのデータの取り出し方法を規定し，部分競合解消処理を具体的に作成し，いくつかの例でプロダクションシステムの実行速度を測定した．その結果，簡単な例では数倍から10倍，複雑な例でも30％の速度向上を達成した．

今後の課題は，他の競合解消戦略への対応や，複雑な応用でのルールの記述性とαメモリの整列キーの関係の調査である．

参考文献

［市瀬 91］“プロダクションシステムを高速化する競合解消”，人工知能研究会　79-7，1991