6J－2

書き換え型プロダクションシステムのための高速マッチングアルゴリズム

藤田 聡, 徳光孝之, 山下雅史, 阿江 忠
(広島大学工学部)

1．はじめに

我々は現在、書き換え型プロダクションシステム(以下RPS)のための処理系を設計・構築中である。RPSは (1) 固定個数のデータの書き換えによって計算を進めていくこと(modify命令のみが右辺で許されている)、および (2) この制約を利用した高速な処理系の実現が期待できることなどの特徴をもつ。以下では RPSのための高速マッチング方式の概要について説明する。

2．仮定

提案方式では書き換え型という制約の他に、以下のことを仮定する。

(1) ルールベース(RB)中のどのルールも、任意に固定されたデータベース(DB, ただし初期DBから到達可能であるもの)において高々ひとつのインスタンスしか生成しない。ただし異なるDBに対しては異なるインスタンスを生成してもよい。

(2) どのルールも、各データのデータ名を変更することはない。

(3) 初期DBにおいてどの名前をもつデータが何個存在するかは、前処理時に予めわかっている。□

(1)はルールが具体的に表現されることを要請しており、提案方式において本質的である。また(2)(3)の各仮定は、通常のPSからみても自然な要請である。

3．例題

次にマッチング方式を具体例により説明する。

[例] 2ルールからなるRBを考える。

(rule 1 (p_1 attr1=<x>, attr2=<y>, attr3=竹) and
(p_2 attr1=<y>, attr2=<z>, attr3=松) and
(p_3 attr1=<z>)
--> modify (p_1 attr2:=松; attr3:=松))

(rule 2 (p_1 attr1=松, attr2=<x>, attr3=松) and
(p_2 attr1=梅, attr2=竹, attr3=<x>) and
(p_4 attr1=<x>, attr2=松)
--> modify (p_2 attr1:=松))

ここで各p_iはデータ名をあらわしており、attrは各データの属性名をあわらす。また<x>,<y>,<z>により変数を、松, 竹, 梅により定数をそれぞれあらわすことにする。このRBは提案する方式では下図のように展開される。ただしDBには名前 p_1, ..., p_4 をもつデータがひとつずつあるものとする(これにより仮定(1)が満足される)。なお二段目の○は属性間の二項関係, ◎は単項関係をそれぞれあらわしている。

さていまDBに次の4つのデータが存在するものとする。

(p_1 attr1=松, attr2=梅, attr3=竹)
(p_2 attr1=梅, attr2=竹, attr3=松)
(p_3 attr1=竹)
(p_4 attr1=松, attr2=松)

このとき、ルール1の条件部が満足され、右辺が実行可能となる。『modify (p_1 attr2:=松; attr3:=松)』の実行に伴って図2の処理がなされる。attr2は二項関係にのみ関与しているため単項関係の再チェック(Reteにおける定数テスト)を要さず、attr3は逆に変数に関する二項関係の再チェックを要しないことに注意。マッチングの結果、それまで満たされていたルール1の左辺が満たされなくなり、ルール2の左辺が新たに満たされる。

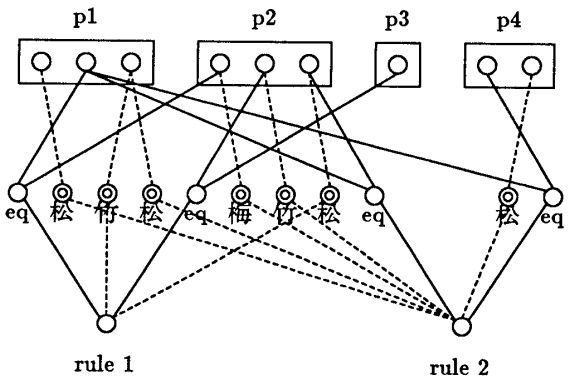


図1 マッチングのためのデータ構造

A Fast Matching Algorithm for Rewriting-based Production Systems, by Satoshi Fujita, Takayuki Tokumitsu, Masafumi Yamashita, and Tadashi Ae, Faculty of Engineering, Hiroshima University

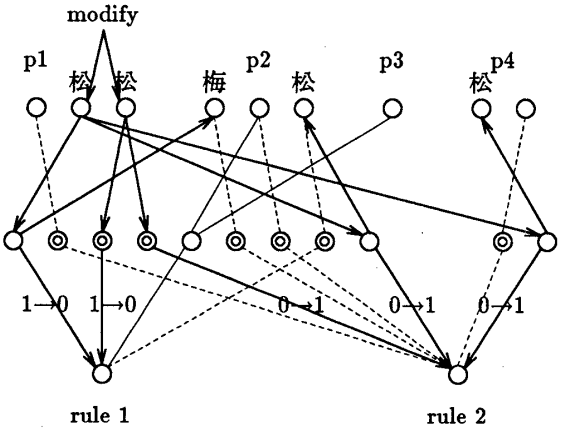


図2　ルール1の実行に伴う処理

4．アルゴリズム

提案するアルゴリズムの概要を以下に示す。

Matching Algorithm for RPS

入力：　DBの変化分，　出力：　競合集合の変化分.

begin

1: ルール実行に伴う属性値の変更

2: 変更された属性に関連する全ての関係の再チェック

3: 充足性の変化した関係に関する全ルールの再評価

end.

アルゴリズム中の『変化分に関連した』属性あるいはルールは、適切な前処理によってあらかじめ検出しておくことができる(これが図1のデータ構造である)。

前節の例からもわかるとおり、このアルゴリズムはRete[1]のような join-based のマッチングアルゴリズムと比較して、特に以下のような点が異なっている。

(1)　属性間関係の再評価に冗長性が少ないこと(←処理の共有率の増加, 属性単位での処理の起動)。

(2)　処理の深さが高々2であり、結合処理に起因する長さ3以上の連鎖を含まないこと。

5．プロトタイプによる評価

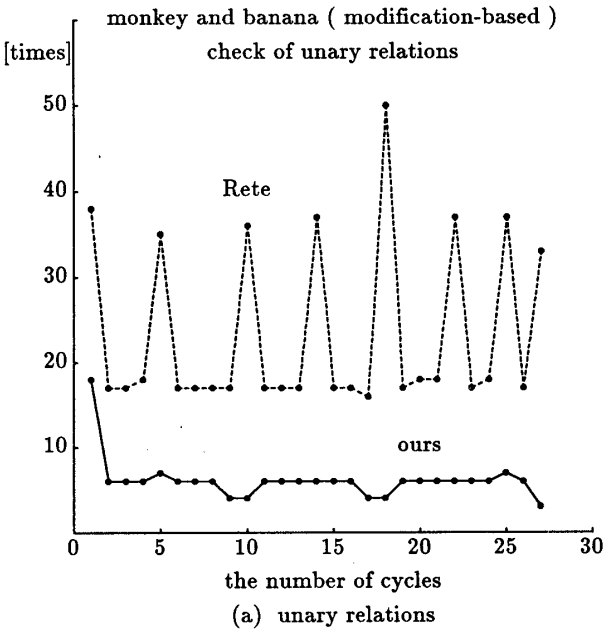
RPSのプロトタイプと Reteアルゴリズム[1]を用いたプロトタイプPSをC言語を中間言語として製作し、ベンチマークテストを行なった(ベンチマークには猿とバナナの問題を用いた)。図3に実験結果を示す。横軸は計算におけるサイクル数、縦軸は各サイクルにおける照合回数をあらわしている。図から、単項関係(a),二項関係(b)とも提案方式の方がReteを上回っていることがわかる。またプログラムの実行時間は、　Rete方式で194*msec*であったのに対し、提案方式では72*msec*であった。

6．あとがき

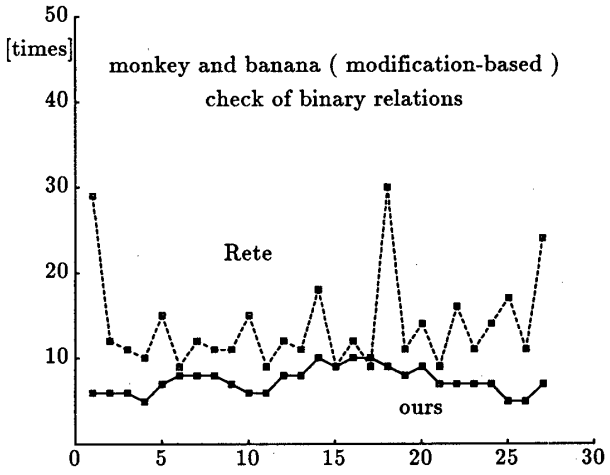
今後、より現実的な問題における効果を評価していく予定である。

[参考文献]

[1] C.L.Forgy, *Artif. Intell.*, 19, pp.17-37 (1982).



(a)　unary relations



(b)　binary relations

図3　ベンチマーク結果