

プロダクション・システムの 並列実行方式

7D-6

長野 ゆかり 松澤 和光

NTT 情報通信処理研究所

1. はじめに

プロダクション・システム(PS)の高速処理をねらいとして複数ルールの並列実行方式¹⁾が提案されている。本稿ではこの方式における並列実行可能な複数ルールの選択法を改良し、さらに先に提案した非同期実行方式²⁾への拡張を行う。

2. 複数ルール実行

2.1 従来の並列実行方式の問題点

複数ルール実行では、その結果が必ずある逐次実行結果と一致しなければならない。しかし、競合集合から次の関係にあるルールを選択すると、いかなる逐次実行結果とも一致しないことがある。

- ① R₁の削除/更新要素がR₂の条件要素と一致
- ② R₁の生成要素がR₂のnot条件要素と一致

この関係をルール1(実行側)はルール2に干渉すると呼び、この関係を調べることを干渉検出と呼ぶ。干渉検出では、どちらのルールを実行側にするかによって干渉する場合としない場合がある。

2ルールの場合は、どちらかが他方に干渉しなければ、干渉しないルールを先に実行したと考えることにより並列実行可能となる。しかし3ルール以上の時は、ルールの実行順を想定して干渉検出を行わなければならない、処理が複雑になる。そこで効率よく選択する方法として、先に選択したルールよりもあとで実行可能なルール、つまり先に選択したルールが干渉しないルールだけを選択する方法がとられている。

しかしこの方法では、干渉検出を行う順序が固定的なため、本来は並列実行可能なルールが選択されないことがある。例えば図1のルールAはBに干渉するが、BはAに干渉しない。そこでAから選択を行う(①)と、並列実行可能なBが選択されない。

また干渉検出には、コンパイル時にクラス等のルール間の関係から静的に行う方法と、PS実行時に照合したWMEを直接比較する動的な方法とある。動的な方法では静的な方法よりも並列実行可能なルールを多く選択できるが、干渉が生じないことが明かであるルール間でも干渉検出を行う無駄がある。

ルールA : if a, c then delete a
 ルールB : if a, b then create d
 ルールC : if not d then create e
 ルールD : if c then modify c, cl

S : 並列実行可能なルール集合, (●) : 干渉検出
 P : 競合集合

①ルール順列A-B-Cの場合 ②ルール順列C-B-Aの場合

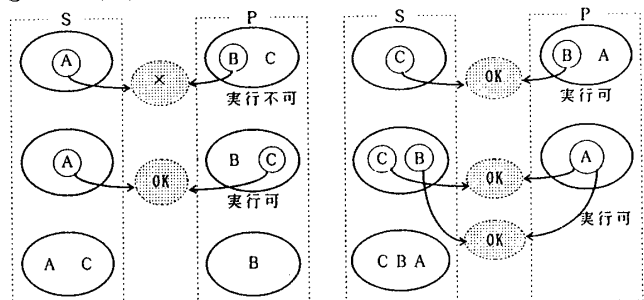


図1 従来法による実施例

2.2 タイプ検出方式の提案

上記問題点を解決するために、以下の考え方から、新しい干渉検出法を提案する。

- (1) 選択したルール順で逐次実行できるか否かのチェックをするのではなく、逐次実行ができるようなルール順を作ることによって並列実行可能なルールを選択する。
- (2) このために、ルール自身の静的な解析情報(タイプ)を各ルールに付加し、これを動的な干渉検出と組合わせて用いる。

本方式ではタイプからルールの干渉関係が判断できるように、not条件要素及び削除/更新の有無によってルールを図2の4タイプに分ける。図3にはタイプの比較によるルール干渉の可能性を示す。このタイプ情報はルール間のクラス関係等と異なりそのルールの形だけで決まるので、簡単に求めることができる。

タイプ	条件部	実行部
	not条件要素	削除/更新
1	含まず	含まず
2	含まず	含む
3	含む	含まず
4	含む	含む

図2 タイプの定義

タイプ検出方式：

次の条件①～③を満たしたルールを競合集合 (CS) から選択し実行ルール集合 (Rb) に格納した後に、条件④を満たしたルールを選択する。

条件

- ① CS中の全てのタイプ1のルール
- ② CS中のタイプ2のルールで、Rbの全てのタイプ2のルールと干渉しないルール
- ③ CS中のタイプ3のルールで、Rbの全てのタイプ3のルールと干渉しないルール
- ④ CS中のタイプ4のルールで、Rbの全てのルールと干渉しないルール

本方式ではまず、タイプ3→1→2→4の順で干渉検出を行うことを仮定する。ここでRbのルールを実行側とした時に干渉検出が必要となる組合せを図4に示した。“-”は仮定した順序ではこのタイプのルールがRbに含まれていないので、生じない組合せである。図4からタイプ1、2、3のルールでは、同じタイプ同志以外では干渉検出の必要がないので、仮定したタイプ順にかまわずどのような順で干渉検出を行ってもよい。

このように本方式では、タイプ1、2、3の間で並列実行可能な全てのルールを選択でき、さらに干渉検出を行う組合せが従来の16通りから6通りに削減される。

図5にルールA～Dでの本方式の動作例に示す。従来の方法では、各2ルールずつ6回の干渉検出が必要であるが、本方式ではタイプ情報の比較から、干渉検出はルールAとDの1回だけで済み、処理効率がよいことがわかる。

○：タイプ情報による比較

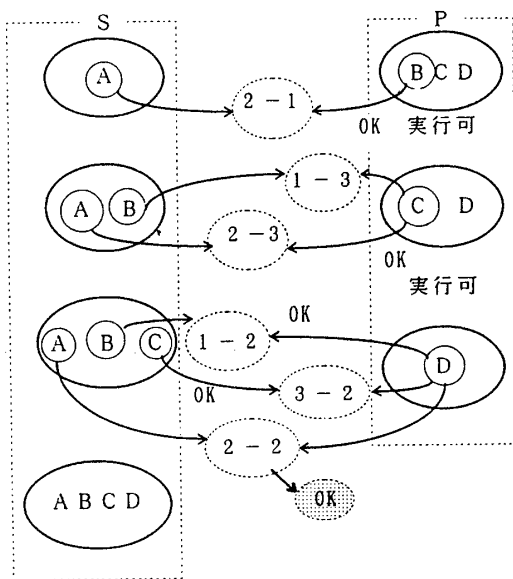


図5 実施例

ルール1 (実行側)

タイプ	1	2	3	4
1	○	×	○	×
2	○	×	○	×
3	×	×	×	×
4	×	×	×	×

ルール
2

○：なし ×：あり
図3 干渉の可能性

Rbのルール

タイプ	1	2	3	4
1	○	-	○	-
2	○	×	○	-
3	-	-	×	-
4	×	×	×	×

CS
の
ル
ル

○：なし ×：あり
図4 タイプ検出方式の干渉検出

3. 非同期実行方式への拡張

非同期実行方式は、各PEが干渉検出を独自に行い、同期をとらずに認知サイクルを実行することによってPSを高速化する方式である。各PEには、他PEの実行したルールを格納するバッファIbを設置し、競合集合から選択した並列実行可能なルールの中から、Ibのルールと干渉しないルールを選択実行する。

この方式の干渉検出にタイプ検出方式を適用するためには、次のような拡張が必要である。非同期実行方式ではIb内のルールが既に実行されているので、ルール実行順を先のように仮定するわけにはいかない。そこでタイプ検出方式で省略した以下のルールの組合せについて干渉の可能性を調べる。

Ibのルール

タイプ	1	2	3	4
1	○	-	○	×
2	○	×	○	×
3	-	-	×	×
4	×	×	×	×

CS
の
ル
ル

タイプ	1	2	3	4
1	○	-	○	×
2	○	×	○	×
3	-	-	×	×
4	×	×	×	×

図6 非同期実行方式での干渉検出

図4の①の組合せでは、IbのルールはCSのルールに干渉する可能性があるが、CSのルールはIbのルールに干渉しないので、CSのルール実行をIbのルール実行よりも先に行うと仮定して選択しても解は矛盾しない。したがって干渉検出は省略できる。②の組合せでは、CSのルールもIbのルールに干渉する可能性があるため、Ibのルールを実行側として干渉検出を行う。

したがって非同期実行方式でルールを選択する場合には、図6のような組合せについて干渉検出を行う。

4. おわりに

今後はタイプ検出方式を用いた非同期実行方式の定量的な評価を行う予定である。

参考文献

- 1) 石田 信学論 J71-D No.3 pp567-576 1988.3
- 2) 長野 松澤 信学会春季全国大会 D-381 1989