ICOT Technical Memorandom: TM-1176

TM-1176

推論システムにおけるルール照合フィルタの並列化について

新谷　虎松（富士通）

May, 1992

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

推論システムにおけるルール照合フィルタの並列化について

Implementing a Parallel Matching Filter for Inference Systems

新谷虎松
Toramatsu Shintani
(株)富士通研究所 国際情報社会科学研究所
FUJITSU LABORATORIES, IIAS

ABSTRACT We propose a new method for paralleling a matching filter that is used for speeding up a matching process in inference engines. To realize the parallel matching filter, we use the parallel logic pr ogramming language KL1 and exploit the framework of an LHS filter. The LHS filter is a Prolog-based matching filter implemented by us. The LHS filter is a fast matching filter using the advantages of a Prol og programming. We effectively use the logical variables, hash indexing, the unification, and the backt racking mechanism for managing object level variables and searching instantiations (that is, results of the matching process). In the KL1 programming, we cant not directly use the advantages. This paper pre sents new mechanisms for managing the variables and searching instantiations in the KL1 based parallel matching filter. The parallel matching filter realizes the full parallelism by using features of KL1.

1 はじめに

OPS5[1]で代表される前向き推論システムの高速化技法として，RETEアルゴリズム[2]やTREATアルゴリズム[6]が良く知られている．これらアルゴリズムは，推論メカニズムにおける照合過程を効率良く実行するためのものであり，McDermottが提案した照合過程のためのルール照合フィルタ[7]の機能を実現している．最近では，推論をさらに高速化するために，これらアルゴリズムの並列化が研究されている．元来，OPS型プロダクションシステムの実行メカニズムにはかなり多くの並列性が内在していることが良く知られている[3]．例えば，D.Kalpらは，RETEアルゴリズムにおける内部メカニズムを並列化した並列化OPS5を試作している[4].

筆者等は，論理型言語の利点を生かした高速なルール照合フィルタ（LHSフィルタと呼ぶ）を実現している[9]．LHSフィルタは論理型言語の利点を利用した論理型言語向きのルール照合フィルタとして特徴付けられる．LHSフィルタは，並列論理型言語であるKL1を用いることにより，効果的に並列化することが可能である．しかしながら，KL1では，LHSフィルタの実現に関連して，Prologで利用したrepeat-fail機能によるバックトラッキングやヘッドユニフィケーションに基づく論理変数束縛チェック機能はなく，新たなLHS節の枠組みが必要となる．本論文では，推論システムにおける並列化照合過程の実現を目指して，KL1に基づくLHSフィルタの並列化について論じる．

2 ルール照合フィルタ

ルール照合フィルタは，照合過程を効率化するためのものであり，照合過程の結果（つまり，インスタンシエーション）を特殊な解釈実行系を介することなく生成するために用いられる．ルール照合フィルタの概念は，McDermott[7]により提案され，(a)Condition Membership，(b)Memory Support，及び(c)Condition Relationshipと呼ぶ3種の高速化に寄与する知識を組み合わせて用いることにより構成される．(a)は，どの条件要素がどのルールに含まれるかについての知識である．(b)は，どのWM要素がどの条件要素を満たしているかについての知識である．(c)は，ひとつのルールのなかでの複数の条件要素間でどのような関係があるかについて知識である．RETEアルゴリズム[2]は，先の(b)と(c)の知識を組み合わせたルール照合フィルタであり，TREATアルゴリズムは，先の(a)と(b)の知識及び新たに(d)としてConflict-set Supportと呼ぶ高速化に寄与する知識を組み合わせたルール照合フィルタに相当する[6]．ここで，(d)は，seed-orderingと呼ばれるjoin演算の順序を実行時に最適化するための知識である．

2.1 LHSフィルタ

LHSフィルタは，ルール照合フィルタに関する(a)，(c)及び(d)の知識を組み合わせたルール照合フィルタの機能を実現する[9]．LHSフィルタでは，ルール照合フィルタに関する(b)のMemory-supportの機能は採用していない（つまり，照合過程において中間結果を保存しない）．採用しない理由は，Prolog処理系が，一般に，このような中間結果を保存・更新するためには多くのオーバーヘッドを必要とするからである．むしろ，LHSフィルタでは，Prologの節の高速な参照機能を効果的に利用することにより，照合時の再照合の効率化を図っている．再照合は，LHS節の本体において，ハッシュインデキシング効果によりWM要素を選択的に参照することにより高速化される．

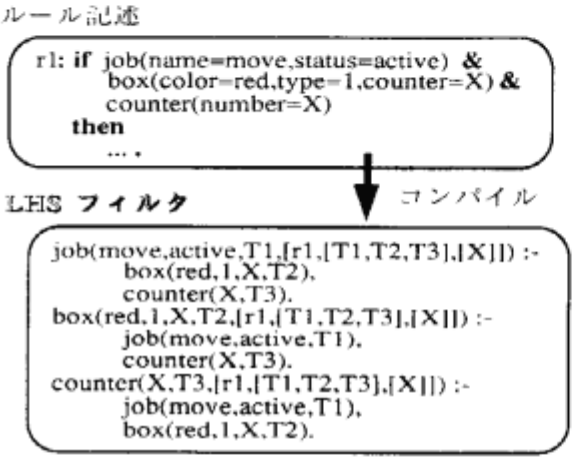

図1. LHSフィルタの生成例

LHSフィルタの概略は図1のように示すことができる．図1は，図上で与えられたルール[8]のLHS（条件部）をコンパイルすることにより，図下で示すLHSフィルタが生成されることを示している．LHSフィルタは，LHS節と呼ばれるPrologのホーン節を用いて実現される．ルールのLHSやWM要素を表すパターンは，スロット記述（"スロット＝値"で構成される）を引数とするPrologの項で表現される．LHS節のヘッドの引数は，スロット値の並び，タイムタグ，及び照合過程の結果であるインスタンシエーションから構成される．ルールのLHSにおける変数は，LHS節ではPrologの論理変数として表現される．スロット値の並びの順は，スロットと対応させるために予め定義される．

ここで，図1で得られたLHSフィルタを用いた照合過程（つまり，インスタンシエーションの探索）を示す．図2はその概略を表している．図2では，WMの変化として，KORE/IEのmakeコマンドを用いたWM要素の生成の例を図示している．makeコマンドは新たなWM要素をWMに付加する一方，WMの変化の分としてLHS節へのProlog queryに変換される．図2において，Prolog queryはLHSフィルタを構成する第1番目のLHS節を選択的かつ高速に呼び出すことになる．これは，普通，Prologの節がファンクタ名（この例では，"job"）と第1引数（この例では，"move"）を用いてハッシュインデキシングされるからである．本体の第1ゴール及び第2ゴールは，ルールを満たすべき他に必要とされるWM要素のパターンを表している．LHS節本体のゴールが成功すると，LHS節のヘッドの最後の引数へ論理変数束縛の情報が節の本体からヘッドへ後向きに伝播される．その結果，呼び出したqueryの最後の引数である"Instantiation"にインスタンシエーション情報が出力される．

照合過程では，普通，複数のインスタンシエーションが生成される．本アプローチでは，Prologのバックトラッキング機能を素直に用いてLHS節に対するqueryの複数解を求めることにより，インスタンシエーションの全解探索を実現する．この複数のインスタンシエーションには，ひとつのルールにおける変数束縛の組み合わせにより得られる場合と，複数のルールごとに得られるものが含まれる．

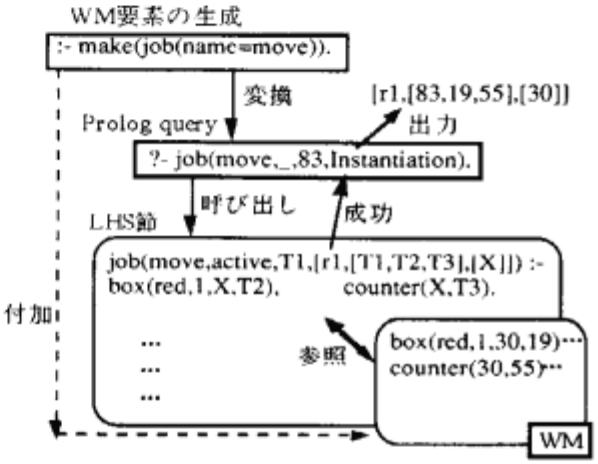

図2. インスタンシエーションの全解探索

3 LHSフィルタの並列化

本章では，並列論理型言語KL1を用いたLHSフィルタの並列化について論じる．KL1では，Prologで利用したrepeat-fail機能によるバックトラッキングやヘッドユニフィケーションに基づく論理変数束縛チェック機能はなく，KL1に基づくLHSフィルタの実装においてこれら諸機能にかわる新たなLHS節の枠組みが必要となる．特に，Prologにおけるバックトラック機能は，インスタンシエーションの複数解を求めるための重要な機能となっている．さらに，KL1では，Prologのようにオブジェクトレベルの変数とメタレベルの変数を同じKL1の論理変数で表現することができない．ここでは，ルールで記述された変数をユニファイ（もしくは，管理）する新たに特別な枠組みを導入する必要がある．オブジェクトレベル変数管理の効率化は，照合過程の高速性を左右する本質的な課題である．本研究では，オブジェクトレベルの変数を効率的に扱うために，越村等により開発された基底項表現に基づくユニフィケーションアルゴリズム[5]を利用する．ここでは，変数環境をベクタ表現をすることにより実現する．

図1で示したLHS節の本体は，WM要素が前もってassertされていることが前提となり，ヘッドユニフィケーション機能やretract機能を用いてWM要素のチェックおよび更新を実現している．KL1では，このようなassert-retract機能に基づくグローバルなデータベース機能はなくWM（および競合集合）をシステムの引数として持って回る必要がある．これにより，Prologで採用したようなインデキシングによる効率的な高速性は犠牲になるが，プログラムの並列性や拡張性は向上する．本研究では，LHS節本体のゴール呼出は，ゴールを並列に実行することによりその高速化を指向する．

KL1を用いて実現したLHS節を便宜上，p-LHS節と呼ぶ．また，p-LHS節により構成されるルール照合フィルタをP-LHSフィルタと呼ぶ．KL1では，先に述べたようにグローバルなデータベース機能がないので，ルール照合フィルタの効率化に関連してRETEアルゴリズムで取り入れたMemory-Support機能を導入することは得策ではない．そこで，p-LHSフィルタは，PrologにおけるLHSフィルタで取り入れたルール照合フィルタとして同様な機能を実現する方向で設計する．

3.1 p-LHSフィルタ

図3は，p-LHS節の概略を示している．図3の例は，図1下における第1のLHS節に対応するものである．p-LHS節は，LHS節で利用されたPrologの利点としての諸機能をKL1の機能を用いて代替し実現する．ここでは，新たにp-LHS節のヘッドにWMを表す引き数が加えられている．例えば，図3のp-LHS節において，第4引き数で示した"[WM_box, WM_counter]"はWMを現す．WMは，照合過程でWM要素をp-LHS節の本体で効率的に参照するために，パターン記述のクラス名ごとに分割されている．分割されたWMは，KL1の提供するプールを用いて管理される．図3のp-LHS節の本体では，boxクラスおよびcounterクラスに関連したWM要素を参照する必要があるので，節のヘッドではこれらWM要素に関連した受け口が用意されている．これにより，p-LHS節がプロセスとして起動した際に受けわたされるWMの通信量も減らすことができ，効率化／並列化が可能になる．

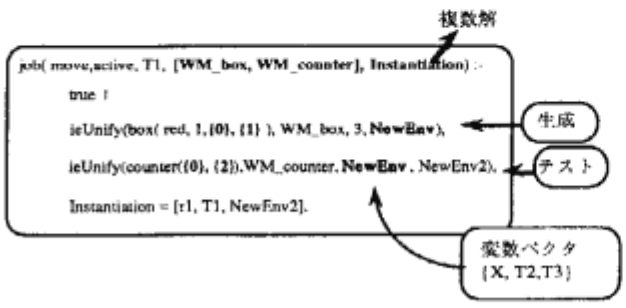

図3. p-LHS節の例

図3のp-LHS節の本体における述語ieUnifyは，Prologのユニフィケーションに相当する機能を提供する．本ユニファイヤは，文献[5]で実現された高速なユニフィケーションアルゴリズムを利用する．ここでは，オブジェクトレベルの変数はベクタメモリ（変数ベクタ）を用いて表現する．述語ieUnifyの第1引数はユニフィケーションで用いられるパターン，第2引数は第1引数とユニファイするために参照するWMを現す．述語ieUnifyの第3引数は，ユニフィケーションで変数束縛の制約条件として用いられる環境を現す．第3引数が具体的な数Nで与えられた場合は，変数束縛環境の初期値として長さNのベクタを与えたことに相当する．ここで，Nは，p-LHS節の本体で束縛される変数の数である．変数は，順に，変数ベクタの位置を表す数字を用いて{0},{1}および{3}として表される．p-LHS節本体で，ルール記述で用いられる共有変数は同じベクタの位置を表す数字が用いられる．例えば，図3のp-LHS節の本体において，{0}はルール記述での共有変数Xに相当する．

図3のp-LHS節の本体において，第1のゴールである述語ieUnifyの第4引数"NewEnv"はユニフィケーションにより得られた変数束縛の結果の情報を保存する変数ベクタのリストを現す．p-LHS節の本体の第2ゴールでは，先に得られた変数束縛の情報"NewEnv"が，新たな変数束縛のための環境として用いられ，ユニフィケーションの結果をNewEnv2へ出力する．p-LHS節では，変数束縛に関連して，パイプライン並列を利用した生成-テスト方式の変数束縛チェックを実現する．これにより，一つのp-LHS節呼び出し（つまり，一つのルール呼び出し）に関連した複数のインスタンシエーションを得ることができる．

図3のp-LHS節のヘッドにおいて，"Instantiation"は，p-LHS節の呼び出しにより得られるインスタンシエーションを表す．このインスタンシエーションは，ひとつのルールから得られる複数のインスタンシエーションに相当する．全てのルールを対象にした複数のインスタンシエーションを得るためには，特別な枠組みが必要である．図4は，このような枠組みを与えるために，p-LHS節の結果（インスタンシエーション）を統合するためのKL1節の例である．このようなp-LHS節の結果を統合するKL1節の集合により，ルール照合フィルタとしてのp-LHSフィルタを実現する．本KL1節を，p-LHS節と区別するためにp-LHSフィルタ節と呼ぶ．

p-LHSフィルタ節は，図3で示されるような複数のp-LHS節を同時に呼び出すための受け口として用いられる．p-LHSフィルタ節は，ルールの条件要素で用いられるクラス名（この例では，"job"）ごとに作られる．つまり，ルールベースでn個のクラス名が用いられるとn個のp-LHSフィルタ節がルールコンパイラにより生成される．これにより，WMの変化の分に即して照合過程を実現する．

図4は，クラス名jobに関連したp-LHS節の例を示している．例えば，クラス名jobに関連したWM要素が作成され

ると，クラス名jobに関連したp-LHSフィルタ節のみが呼び出される．その後，クラス名jobパターンを持ったルールからインスタンシエーションを得るためにクラス名jobに関連したp-LHS節が呼び出される．これは，クラス名jobに関連したインスタンシエーションを全解探索するために各ルール（つまり，p-LHS節）に対してジョブを依頼していることに相当する．

```
job(Name, Status, Tag, WM, Instantiations) :- true |
      takeWM(box,WM, Box_WM, WM2),
      tekeWM(counter,WM2, Counter_WM, WM3),
      job1_1(Name,Status, Tag, [Box_WM,Counter_WM], Inst1),
      job1_2(Name,Status, Tag, [Box_WM], Inst2),
      job1_3(Name,Status, Tag, [Counter_WM], Inst3),
        ...
      merge([Inst1,Inst2,Inst3], Instantiations).
```

図 4．インスタンシエーションの統合

p-LHSフィルタ節の本体では，ルールに現われるパターン記述ごとにファンクタ名がユニークに決定されたp-LHS節のヘッドパターンが並べられる（つまり，図3で示されるようなp-LHS節のヘッドのファンクタ名はルールの条件要素ごとにユニークに決定される），実際，図3のp-LHS節のヘッドのファンクタ名jobは，図4のp-LHSフィルタ節における本体で示すようなjob_1として定義される．これは，p-LHS節を並列に呼び出すための工夫である．ゴールの実行（つまり，p-LHS節の呼び出し）結果は，KL1の組み込み述語であるmergeにより結合される．これにより，各ゴールは，通信を極力避けることにより効率的に実行される．

4 おわりに

実際のKL1プログラミングでは，KL1節の本体で使われる変数の数やゴールの数が制限されており，p-LHS節やp-LHSフィルタ節の実現は，図３で示したp-LHS節や図４で示したp-LHSフィルタ節の例より複雑である．ここでは，引き数をベクタ化したり，p-LHS節の呼び出しを再帰的に呼び出す工夫をすることにより，KL1プログラミングでの制限を回避している．さらに，効率的なKL1プログラミングを実現するためには，ゴールやデータのPE間の受け渡しに関連してMRB（多重参照ビット）を考慮したプログラミングが必要である．ここでの工夫は，KL1プログラミング全般において考慮すべきものであり，本論文では，p-LHSフィルタを論じる上で本質的でないので省略した．

p-LHSフィルタにおいて，本論文では主にp-LHSフィルタの効率性の向上を主目的とした．本アプローチは，並列度を考慮した実現手法を導入したことにより，p-LHSフィルタの枠組み自身は，台数効果を期待できる．しかしながら，実際のMulti PSIをベースにした現KL1 処理系では，PE間の通信のコストが非常に高く，また，p-LHSフィルタの内部実行メカニズムの粒度が小さすぎるので，負荷分散による台数効果が得られていない．これは，負荷分散という観点からは，粒度の大きい仕事を分散する必要があることを示している．例えば，複数のルールベース毎に負荷を分散する等の工夫が必要である．ルールベース毎の負荷分散方式は今後の課題である．

Prologを用いたLHSフィルタは，Prologの利点である論理変数束縛機能，バックトラキング機能および節のインデキシング機能の効率性を十分に取り入れたことによりその高速性を実現した．一方，KL1を用いたp-LHSフィルタの実現には，KL1の並列論理型言語としての制約によりこれらPrologでの利点を用いることはできない．本研究では，照合過程のさらなる高速化のために並列化照合過程の必要性を背景にして，新たな視点でKL1を用いたLHSフィルタの並列化を指向した．プロトタイプとしてp-LHS節およびp-LHSフィルタ節により構成されるp-LHSフィルタを実現した．並列化照合過程はp-LHSフィルタにより実現する．今後の課題は，複数のプロセッサを効果的に用いるための実際的な負荷分散方式を取り入れることによる，照合過程の効率化および高速化である．

尚，本研究は第５世代コンピュータプロジェクトの一環として行なわれたものである．

参考文献

[1]Forgy, C.L.: OPS5 User's Manual, CMU-CS-81-135, 1981.

[2]Forgy, C.L.: Rete :A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, vol.19, , pp. 17-37, 1982.

[3]Gupta, A., et al.: Parallel algorithms and architectures for rule-based systems, in International Symposium on Computer Architecture, pp.28-37, 1986.

[4]Kalp, D.,e. al.: Parallel OPS5 User's Manual, CMU-CS-88-187, 1988.

[5]越村三幸, 藤田博, 長谷川隆三: KL1上のユニフィケーションプログラムとその評価, ICOT TM-975, 1990.

[6]Miranker, D.P.: TREAT:A Better Match Algorithm for AI Production Systems, in AAAI-87, pp.42-47, 1987.

[7]McDermott, J., A. Newell, J. Moore: The Efficiency of Certain Production Implementations, in Pattern Directed Inference Systems, Academic Press: pp. 155-176, 1978.

[8]Shintani, T.: A Fast Prolog-based Production System KORE/IE, in the Fifth International Conference and Symposium on Logic Programming, MIT Press, pp.26-41, 1988.

[9]新谷虎松: prologにおけるプロダクション照合フィルタの高速化, 情報処理学会論文誌, vol.32, No.1, pp. 20-31, 1991.