Ｐｒｏｌｏｇにおけるルール照合フィルタの改良について

富士通㈱国際情報社会科学研究所
新谷虎松

１．はじめに

OPS5[Forgy 81]で代表される前向き推論型プロダクションシステムの高速化技法として，RETEアルゴリズム[Forgy 82]やTREATアルゴリズム[Miranker 87]が良く知られている．これらアルゴリズムは，プロダクションシステム（ＰＳ）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[McDermott 78]が提案した照合フィルタの概念を用いてよく説明できる．実際のインプリメンテーションでは，一般に，照合フィルタはルールコンパイラーを用いて，ルールの条件部からある種のネットワークを構成することにより実現される．　さらに，RETEネットワークでは，ネットワークのノードを共有することによりネットワークのメモリスペースの削減と照合過程の高速化を図っている．

一方，Prolog上でこのようなネットワークに基づくフィルタを効率的に実現することは困難である．そこで，筆者らは，Prologの利点を生かした照合フィルタ生成のためのルールコンパイラを提案した．本コンパイラを組み込んだプロダクションシステムKORE/IEの高速性はOPS5よりすぐれた結果を示した[新谷 87，Shintani 88]．

本アプローチでは，ルールの条件要素毎にLHS節と呼ばれるホーン節をひとつ生成する．これによりPrologのユニフィケーション機能やハッシュ・インデキシング機能を十分に利用した効率的な照合フィルタ（この照合フィルタのことをLHSフィルタと呼ぶ）の生成に成功している．しかしながら，LHSフィルタは，条件要素毎にLHS節を生成する必要性からメモリスペースの点で問題があった．本講演では，特に，メモリスペースを考慮したLHSフィルタの改良について論じる．

２．ＬＨＳフィルタ

照合フィルタは，ＰＳにおける認識－行動サイクルでの競合集合を特殊な解釈実行系を介することなく生成するためのものである[McDermott 78]．このフィルタは，①Condition-membership，②Memory-support，及び③Condition-relationshipと呼ぶ３種の知識を組み合わせて用いることにより構成される．①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．②は，どのワーキングメモリ（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．③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．例えば，RETEアルゴリズムは，先の②と③の知識を組み合わせたフィルタに相当する．また，TREATアルゴリズムは，先の①と②の知識及び新たに④としてConflict-set-support知識を用いたフィルタに相当する[Miranker 88]．④は，seed-orderingと呼ばれるαメモリ間のjoin演算順序を最適化するための知識である．

LHSフィルタでは，②のMemory-supportは導入していない（つまり，照合過程において中間結果を保存しない）．導入しない理由は，Prolog処理系が，一般に，このような中間結果を保存するためには多くのオバーヘッドを必要とするからである．そこで，LHSフィルタは，①のCondition-membership，③のCondition-relationship，及び④のConflict-set-support知識を組み合わせた照合フィルタとなっている．

LHSフィルタの詳細は文献[新谷 87，Shintani 88]に詳しいが，概略を示すと図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記述をコンパイルすることにより，図下で示すLHSフィルタを生成することを示している．LHSフィルタにおける個々の節はLHS節と呼ばれる．図で示すようにLHS節はルールの条件要素の数だけ生成される．Ｖｉは，スロット記述Ｓｉからコンパイル時に具体的に決定される複数のスロット値の並びを表す．Tiはマッチする WM要素のタイムタグを表す（変数表現される）．LHS節は，WMの変化に伴い呼び出され，その結果（LHS節のヘッドの第一引数が具体化された結果），第一引数からインスタンシエーションが得られる．

３．メモリスペースの削減

図１で示したように，LHSフィルタでのLHS節の数は条件要素の数だけ生成され，メモリスペースを必要とする．

Optimizing Matching Filter in Prolog-based Production System

Toramatsu SHINTANI
IIAS-SIS, FUJITSU LIMITED

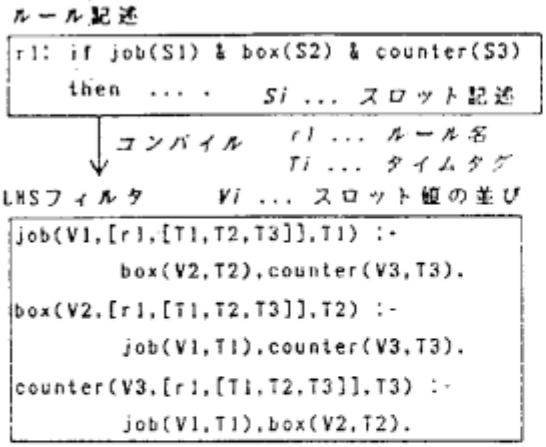

図１．LHSフィルタの例

一方，これはWM要素の変化に伴うそれ以後の処理（LHS節の呼び出し）を高速化するために，Prologのインデキシング機能を利用するためのものであった．例えば，クラス名（つまり，パターン名）boxに関連したWM要素の変化は，直接，図１の２番目のLHS節を呼ぶことになる．この呼び出しは，Prologのインデキシング機能により高速に実行される．

メモリスペースの削減のためには，条件要素毎にLHS節を作らない必要がある．これは，LHS節ヘッドに対するPrologインタプリタ（例えば，C-Prolog）のインデキシング効果を犠牲にすることになる．しかしながら，Prologコンパイラ（例えば，Quintus Prolog）の利用や照合フィルタとして旧LHSフィルタと同様な機能を実現することにより実用的な高速性が期待できる．図２にメモリスペースを考慮したLHSフィルタの概略を示す．

```
lhs_clause([[V1,job,T1],
            [V2,box,T2],
            [V3,counter,T3]],
           [r1,[T1,T2,T3]]) :-
        job(V1,T1),
        box(V2,T2),
        counter(V3,T3).
```

図２．ﾒﾓﾘｽﾍﾟｰｽを考慮したLHSフィルタ

図２におけるLHSフィルタは，Prologコンパイラを用いることにより，LHS節ヘッドの第一引数に対するインデキシング効果を得ることができる．本アプローチの特長は，ひとつのルールにひとつのLHS節を対応させたことである．これによりメモリスペースを削減する．削減量はルールの条件要素の数に依存するが，平均（条件要素が３つとする）約60%のメモリスペースを削減する．

第一引数にあるリスト（Lと呼ぶ）は，WMの変化に即した照合過程を実現するためのものである．リストの長さは，ルールベースで用いられるクラス名の数に相当する．また，リスト要素の位置はクラス名毎に決定される．このようなリストLを構成するために，KORE/IEのliterarizeコマンドの情報を用いる．literalizeコマンドは，ルールベースで用いられるパターンを規定し，効率的なLHSフィルタを構成するための情報を提供する．例えば，本例では，パターンjob，box，及びcounterがこの順でliteralize宣言されている．本LHS節ヘッドの第二引数は，図１で示したインスタンシエーションと同じである．

本LHS節を呼び出すためのqueryは，図３で示される．

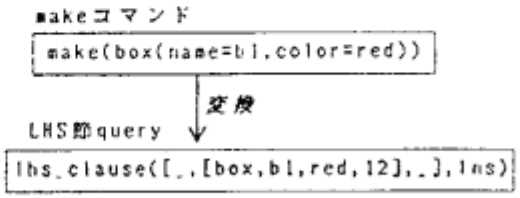

図３．LHS節へのquery例

図３で示すように，例えば，本queryはWMを変化させるコマンドmakeにより生成され，LHS節を呼び出す．この時，パターンboxに関連した（WM要素の変化に関連した）インスタンシエーションInsが計算される．これは，先に示したリストLの２番目の位置だけが具体化されたqueryを生成し用いたからである．　これにより，第２節で述べたConflict-set-support照合フィルタの機能を実現する．さらに，LHS節の本体に必要な条件要素を列挙したことにより，図１のLHSフィルタがもつその他のフィルタ機能も実現する．

４．　おわりに

本論文では，照合フィルタとしてのLHSフィルタの改良について述べた．本アプローチでは，Prologコンパイラを前提としたLHSフィルタを提案した．これにより，高速性を保存し，かつメモリスペースを平均約６０%削減することができた．　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

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

[Forgy 82]C.L.Forgy: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Matching Problem, AI 19,pp.17-37,1982.

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

[McDermott 78]J.McDermott and et.al: The Efficiency of Certain Production System Implementations, In PDIS, Academic Press, pp.155-176,1978.

[Shintani 88]T.Shintani: A Fast Prolog-based Production System KORE/IE, Proc. of LP88,MIT Press,pp.26-41,1988.

[新谷 87]新谷:推論エンジンKORE/IE -反駁メカニズムに基づく高速な推論エンジン-, LPC87,pp.233-242,1987