知識ベース指向の並列推論処理システム*

北上 始, 横田治夫, 服部 彰
(富士通株式会社)

1. はじめに

人間の知的問題解決を計算機上で実現するためには，知識ベース管理機構と推論機構を合わせもつ知識ベース指向の推論処理システムが必要であると言われているが，知識ベースの大規模化や推論処理における組み合わせ爆発の傾向が強くなるにつれ，システムの高速処理が要求されるようになっている。

このようなシステムの高速処理に答えるために，我々は，並列論理型言語GHC 等により，知識ベース指向の推論処理システムを並列化する研究を行ってきている。これまでに，我々は，メタプログラミング の考え方に基づく GHCプログラミング において，GHC の単一代入制限による実現上の問題を，知識ベースに$ 変数を導入する方法で解決してきた。(1) 。

本稿では，さらに，知識ベース指向の並列推論処理システムの拡張性を明らかにするために，カットシンボル 処理や制約論理プログラミング へ展開する方法について述べる。

2. 並列プログラミング

本システムは，知識ベースを持つと同時に，全解探索を行う並列推論処理システムであり，GHC により，以下のように並列プログラミング されている。

```
bagof-solve( [] ,VL,Binf,His,Result):- true |
   『解くべきゴール列が [] なので，このパスの探索を終了する』.
bagof-solve( [true] ,VL,Binf,His,Result):-true |
   『解くべきゴール列が "true" なので，このパスの探索を終了する』.
bagof-solve( [ P| Q] ,VL,Binf,His,Result):-
   Q ≠ [] |
     bagof-solve( [ P] ,VL,Binf,His,Result1),
     and-solve( Q,Binf,Result1,Result).
bagof-solve( [ P] ,VL,Binf,His,Result):-
     otherwise |
     clause-stream( P, ClauseStream),
     or-solve( ClauseStream,VL,Binf,His,Result).
```

"bagof-solve " の第 1 引き数には，解くべきゴール列が入力される。ここでは，ゴール列をリストで表現している。第 2 引き数には，解かれたゴール列の出力変数がリスト入力される。第 3 引き数には，推論中に $ 変数にバインドされた情報がリスト出力される。第 4 引き数には，推論中に使用されたホーン節が使われた順にリスト出力される。第 5 引き数には，推論結果がストリーム として返される。

前記の第 1 番目と 2 番目のプログラム は，ある推論パスにおける推論処理の停止を行う処理である。第 3 番目のプログラム は，AND ゴール列を左から順に解いていくプログラム である。この時，ANDゴール間を共有する$ 変数についても，先に処理されたゴールの$ 変数値を他のゴールに伝播させるために，第 3 番目のプログラム で，変数 "Result1 " を "bagof-solve " から "and-solve " に渡している。第 4 番目のプログラム は，知識ベース検索 "clause-stream " と検索結果の分析 "or-solve" を行う処理である。 "clause-stream " は GHC のGCの低減や大規模な知識ベースアクセス を高速化する為に RBU (2) 等で実現されている。以上，これらのプログラム は，複数解を並行に探索する為に，知識ベースから検索された複数のホーン節を次々と処理するプログラミング になっている。

3. カットシンボル の処理

前述したプログラミング スタイルをさらに押し進めると，カットシンボル の入った知識ベースに対する並列推論処理への拡張も容易に可能になる。

ここでは，簡単のため，カットシンボル は，1つのホーン節に一箇所だけ利用されるものと限定し，条件部 "P, ! ,Q" を，以下のような "ifthen(P,Q) " で表現する。

```
children( $1, $2):-
    children( $1, [] , $2).
children( $1, $2, $3):-
    ifthen( father( $1, $4),
            children( $1, [$4 | $2] , $3)).
children( $1, $2, $2).
```

この知識ベースの例では， " children($1, $2)" に対する質問処理で，父親 " $1 " に対する子供を全て見つけ，その結果をリストで " $2 " に返すことを期待している。この例で示されるような "ifthen(P,Q) " を処理するための並列プログラム は，前節の "bagof-solve " の第 3 番目と第 4 番目との間に，以下のようなプログラム を追加することによって実現される。

```
bagof-solve( [ifthen(P,Q) ] ,VL,Binf,His,Result):-
     true |
     bagof-solve( P, VL, Binf, His, Result1),
     then-part( Q, Binf, Result1, Result).
```

" then-part" は， " P" の分析結果を使って， "ifthen (P,Q) " の " Q" を分析するような以下のプログラム である。

```
then-part( Q, BindInf, [ Result1 | Result2 ] ,
  Result):- true |
     and-solve( Q, Binf, [ Result1] ,Result2),
     『Result2 が [] ならば Result に [fail] を返し，
     [] 以外の場合はResultにResult2 を返す』.
then-part( Q, BindInf, [] , Result):-
  true | Result = [] .
```

Knowledge Base Oriented Parallel Inference System

H.Kitakami, Haruo Yokota, and Akira Hattori
Fujitsu Limited

* 本研究は，第五世代コンピュータプロジェクトの一環として行われた。

さらに，前述の "or-solve" のﾌﾟﾛｸﾞﾗﾑ に，以下のようなﾌﾟﾛｸﾞﾗﾑ を追加することによって実現される。

```
or-solve(  [ [(Head:-  [ifthen(Cond,Body) ] ,
  Inf ]  | ClauseList] ,VL, Binf, His,Result):-true |
    「$ 変数の代入処理により，検索結果得られた
     $ 変数のﾊﾞｲﾝﾄﾞ情報Inf で変数ﾘｽﾄ VLを書き換え，
     NewVL を作成」,
     append( Inf, Binf, NewBinf),
     append( His, [(Head:-  [ifthen(Cond,Body)
     ] ] , NewHis),
     bagof-solve( [ifthen(Cond,Body) ] ,
          NewVL, NewBinf, NewHis, Result1),
     next-solve( ClauseList, VL, Binf, His,
          Result1, Result).
```

このﾌﾟﾛｸﾞﾗﾑ 内の "bagof-solve " により " ifthen(Cond,Body)" が分析されると，" next-solve " では，"Result1 " が [] のとき，次の "ClauseList" を，"or-solve" で分析し，それ以外のとき，"ClauseList" を捨て，枝刈りを行う。

4．制約論理ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ

制約論理ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ は，推論中に知識ベースから数式を抽出できる様に，前述の "bagof-solve " の引き数を増やし，さらに制約処理系を付け加えることによって，容易に実現可能である。制約処理系は，連立一次方程式に帰着されるものに限定している。これにより最終段階で行列計算が必要になる。行列計算の並列化は，行列の各要素にGHC プロセスを割り付け，ｶﾞｳｽｼｮﾙﾀﾞﾝ の計算ｱﾙｺﾞﾘｽﾞﾑを適用することにより容易に実現される。

ここでは，静電界の例題を採り上げ，並列推論処理システムを核とする制約論理ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ の性能について説明を行う。

4.1 静電界の問題

図１に示される二次元矩形領域に於いて，この領域内のある点の電位 "A " がある条件を満足するように，上端の電位 "X " を決定する問題について説明する。この領域はメッシュで分割されているが，各格子点の電位は，Liebmannの5 点近似法で定められるものとする。

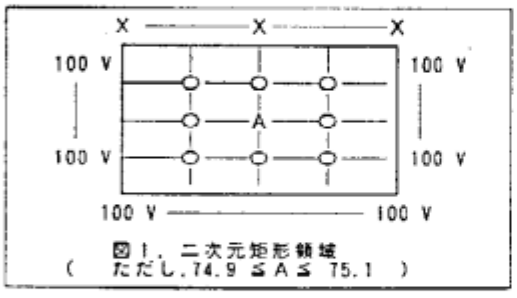

図１．二次元矩形領域
（　ただし，74.9 ≦ A ≦ 75.1　）

この問題を表わす知識ベースは，図２ように，$ 変数で表現される。代数式の等号は，簡単化のため，$ 変数の単一化命令 "unify " で表現することを避け，不等号と同じく，直接，算術記号 " =" を使っている。

```
design(  [$1 | $2] , $3, $4 ):-
    member( $3,  [-1,0,1] ),
    laplace( [$1 | $2] ), 74.9 ≦$4, $4≦ 75.1 .

member( $1,  [$1 | $2] ).
member( $1,  [$2 | $3] ):-  member( $1, $3).

laplace( [ $1, $2 ] ).
laplace( [ $1, $2, $3 | $4] ):-
    laplace-vec( $1, $2, $3),
    laplace( [ $2,$3 | $4] );

laplace-vec( [ $1, $2 ] ,
             [ $3, $4 ] , [ $5, $6 ] ).
laplace-vec( [ $1,$2,$3 | $4] ,
             [ $5,$6,$7 | $8] ,
             [ $9,$10,$11 | $12 ] ):-
    $10 + $2 + $5 + $7 - 4×$6 = 0,
    laplace-vec( [ $2,$3 | $4] ,
                 [ $6,$7 | $8] , [ $10,$11 | $12 ] ).
```

図２．知識ベース

4.2 システムの性能

図２の知識ベースに対して，次のような問い合わせによる性能測定を行った。

```
?- constraint-solve(
   design(  [ [ $900, $900,$900,$900,$900] ,
              [  100, $911,$912,$913,  100] ,
              [  100, $921,$999,$923,  100] ,
              [  100, $931,$932,$933,  100] ,
              [  100,  100, 100, 100,  100] ] ,
          $900, $999)).
```

変数 "$900" は，計算すべき上端の電位であり，変数 "$999" は，矩形領域内の中心の電位である。

測定の結果，行列計算の部分は，GHC が８台のとき，１台に比べて６倍近い並列性能を得ている。また，制約論理ﾌﾟﾛｸﾞﾗﾐﾝｸﾞｼｽﾃﾑ としては，RBU に１台のプロセッサを割り当て，GHC に割り当てるプロセッサの台数を増やすことにより，GHC が９台のとき，GHC が１台に比べて９倍近い並列性能を得ている。

5．おわりに

知識ベース指向の並列推論処理システムの実現方式であるﾒﾀﾌﾟﾛｸﾞﾗﾐﾝｸﾞ の性質を使い，容易にｶｯﾄｼﾝﾎﾞﾙ の処理が可能であることを示した。また，本システムを中核とした制約論理ﾌﾟﾛｸﾞﾗﾐﾝｸﾞｼｽﾃﾑ が容易に実現できることも示した。さらに，この制約論理ﾌﾟﾛｸﾞﾗﾐﾝｸﾞｼｽﾃﾑ の並列性能を測定した結果，良好な台数効果が得られた。

【　謝辞　】本研究開発に当たり，有益なコメントを下さったICOTのKBM のメンバーの方々及び富士通研究所の林人工知能研究部長に感謝致します。

【参考文献】

(1) 北上，横田，服部：知識処理向き並列推論ﾒｶﾆｽﾞﾑ，情報処理38回全国大会，pp.1011-1012，1989.

(2) 横田，北上，服部：知識ﾍﾞｰｽ 指向並列処理ｼｽﾃﾑ，情報処理学会研究会，計算機ｱｰｷﾃｸﾁｬ 81-2，1990.