項インデクスによる知識検索の高速化

7H-4

横田 治夫、 北上 始、 服部 彰
富士通株式会社

1．はじめに

我々は、第５世代コンピュータプロジェクトの一環として、知識ベース指向で並列に問題解決を行なう方法を研究している。現在、知識検索システム（ＲＢＵ）［1］と並列論理型言語（ＧＨＣ）［2］を組み合わせて並列問題解決システムを実現する方法を検討中である［3］。本論文では、その問題解決に利用される知識検索システムの高速化の手法として、項に対するインデクスを提案する。また、試作したプロトタイプを測定し、検索におけるインデクスの効果および更新におけるインデクス維持のためのオーバヘッドについて評価する。

2．背景

並列問題解決システムでは、知識ベースを有効に利用して、知識の要素を効率よく並列に取り扱うことが重要である。ＧＨＣは、ストリームを使った並列処理を記述するのには強力な言語であるが、知識ベースのようなグローバルなデータを扱うのは得意ではない。そこで、知識ベースを効率よく検索、更新できるシステムを接続して利用することが必要となってくる。

ＲＢＵは、知識の要素を項で表現し、項関係と呼ばれる項の集合から、単一化（ユニフィケーション）を導入した関係演算を使って知識ベースを検索するシステムである。項は、ＧＨＣで取り扱われる構造体そのものであるため、２つのシステムの相性は非常に良い。一階述語論理における節や、プロダクション・システムで利用されるプロダクション・ルール、あるいは問題解決の途中で生成される中間状態などを項関係に格納して、ＲＢＵのコマンドを使ってＳＬＤ演繹やプロダクション・システムの実行を行う方法が提案されている［1，3］。

ＲＢＵ自身を並列に実行することも重要な課題であるが、まず逐次でも高速に検索・更新処理が可能となる機構を実現することが必要である。ここでは、専用のハードウェアを想定しないソフトウェアによる高速化の手段として、ＨａｓｈとＴｒｉｅを使ったインデクスを提案する。ここで、Ｔｒｉｅ構造とは、先頭から見て要素が同じ部分をまとめた一種の木構造である［4］。

Term Indexing for Fast Knowledge Retrieval
Haruo Yokota, Hajime Kitakami, Akira Hattori
FUJITSU LIMITED

3．インデクスの実現方式

まず、項を線形表現に展開して、その先頭の要素によってＨａｓｈをかける。このとき、先頭の要素が等しく、同じＨａｓｈ項目に落ちる項の集合を、Ｔｒｉｅ構造にする（図１）。こうすることにより、先頭の要素が同じで構造の似た項が多くある場合でも、効率よく検索することができる。特に、Ｔｒｉｅの上で直接単一化を行なうことにより、単一化のための検索条件との比較操作とバックトラック処理の回数を減らすことができる。

項の平均要素数をＭ、項の数をＮとした時、インデクスを用いない場合の最悪の比較回数はＭ×Ｎとなる。これは、検索対象のすべての項が、最後の要素を除いてすべて等しい場合と考えることができる。この場合、Ｔｒｉｅを用いると比較の回数をＭ＋Ｎに減らすことができる。このときのインデクスの形態は、図２のタイプＡのようになる。先頭の要素が異なる場合には、タイプＢようなインデクスの形態となり、Ｈａｓｈの効果で比較の回数およびバックトラックの回数がＨａｓｈ項目につながれた部分に絞られる。Ｔｒｉｅの効果が一番大きく現われるのは、Ｔｒｉｅの形態がバランスした木構造になるタイプＣのような場合で、要素を比較するたびに絞りこまれるため、比較回数はＭに非常に近くなり、バックトラックの数もわずかになる。

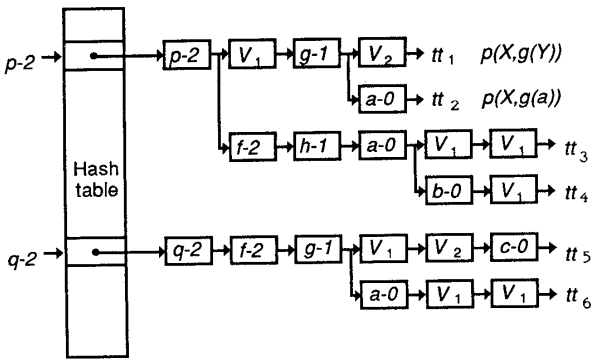

図１． ＨａｓｈとＴｒｉｅを使ったインデクス

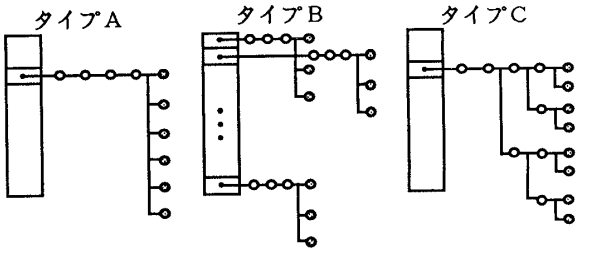

図２． インデクスの形態

4．評価結果

前述したインデクスの効果を、今回試作した知識検索システムのプロトタイプ（逐次版）で検証した。

まず、タイプAのインデクスができるような項関係を生成し、インデクスを張らなかった場合、インデクスを張った場合の検索時間を測定した。また比較対象として同様の節集合に対するＱｕｉｎｔｕｓ－Ｐｒｏｌｏｇインタプリタの検索時間も計測した。この結果を図３に示す。このような項関係に対しては、インデクスを張ってもＨａｓｈの効果はないが、Ｔｒｉｅの効果によって要素間の比較回数

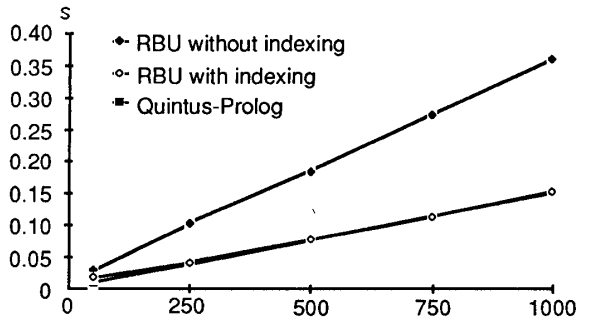

図３．　タイプAに対する検索時間

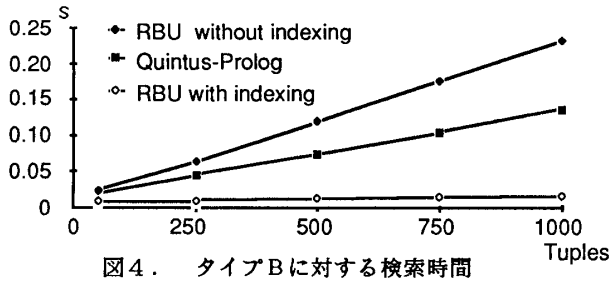

図４．　タイプBに対する検索時間

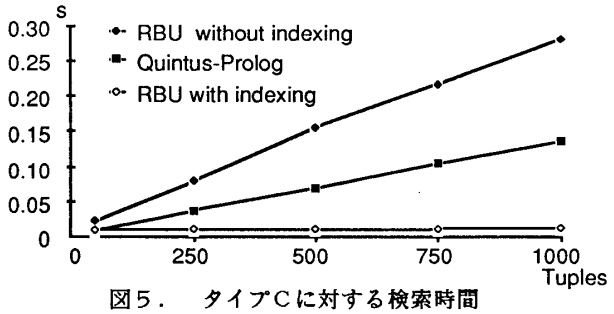

図５．　タイプCに対する検索時間

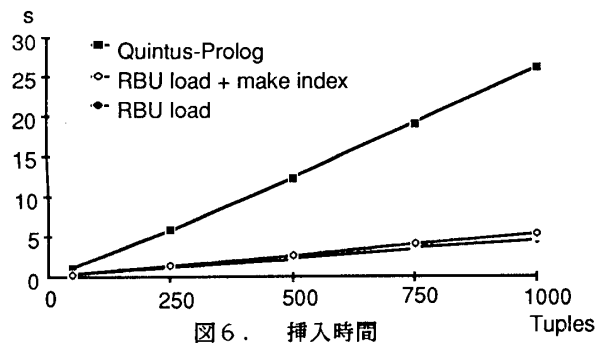

図６．　挿入時間

はＭ＋Ｎになる。このため、どの場合もタプルが増えるに従って検索時間がかかるが、インデクスを張っておくことにより傾斜が緩やかになる。特に、項が複雑な（Ｍが大きい）場合に効果がある。

図４は、タイプBのインデクスができるような項関係に対する同様の測定結果である。これは、Ｈａｓｈによる絞り込みの効果を示しており、項の数が増すことに対する検索時間の増加がほとんどないことが分かる。

一方、タイプCのインデクスができる項関係を使った測定結果を示したのが図５である。この場合インデクスはＴｒｉｅの効果だけとなるが、分岐によりバックトラックの数が大幅に削減されるため、検索時間がタプル数にまるで依存していないことが分かる。Ｈａｓｈ表の大きさにもよるが、タイプＢより速くなっている。

この他、更新処理におけるインデクス維持のオーバヘッドを評価するための測定も行なった。本インデクスはセル構造により実現され、構造が比較的単純であるため、挿入、削除操作を高速に行なうことができる。図６にＰｒｏｌｏｇとの挿入速度の比較と、インデクス作成にかかる時間を示した。インデクス作成が挿入の時間に対してわずかの割合でできることが分かる。

5．おわりに

知識ベース指向の問題解決機構として、知識検索システムを並列論理型核言語で制御する方法は、比較的素直なアプローチであると思われる。検索・更新処理は、問題解決処理の大きな部分を占めるため、検索・更新処理の効率化を図ることが重要である。今回の評価により、ＨａｓｈとＴｒｉｅによるインデクスが検索の効率化に有効であり、更新においてもオーバヘッドがわずかであることがわかった。今後ＲＢＵ処理の並列化についても、研究を進めていく予定である。

謝辞

日頃、御指導御助言を頂くＩＣＯＴ伊藤第三研究室長、富士通研究所、林人工知能研究部長、ならびにＩＣＯＴのＫＢＭメンバーに感謝いたします。

参考文献

[1] H. Yokota and H. Itoh, A Model and Architecture for a Relational Knowledge Base, Proc. of the 13th International Symposium on Computer Architecture, pp2-9, 1986.

[2] K. Ueda, Guarded Horn Clauses, ICOT Technical Report TR-103, 1985.

[3] H. Yokota, H. Kitakami and A. Hattori, Knowledge Retrieval and Updating for Parallel Problem Solving, ICOT Technical Report TR-380, 1987.

[4] D. E. Knuth, The Art of Computer Programming, Vol 3, Sorting and Searching, Addison-Wesley, 1973