

TM-0528他

情報処理学会第37回全国大会
(知識ベースマシン関係)

September, 1988

©1988, ICOT

ICOT

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

(03) 456-3191-5
Telex ICOT J32964

Institute for New Generation Computer Technology

- TM 0528 項インデックスによる知識検索の高速化
横田 治夫、北上 始、服部 彰(富士通)
- TM 0529 Strongly Reduced Systems of Equations
Philippe DEVIENNE
- TM 0545 知識ベースマシンMu-X(1)ー並列制御方式ー
酒井 浩、仲瀬明彦、柴山茂樹、物井秀俊、伊藤英則
- TM 0546 知識ベースマシンMu-X(2)ーキューリ解析方式ー
仲瀬明彦、酒井 浩、物井秀俊、伊藤英則、三友雄司
- TM 0547 知識ベースマシンMu-X(3)ー表現言語から項型言語への変換ー
武脇敏晃、酒井 浩、柴山茂樹(東芝)、物井秀俊、森田幸伯
- TM 0548 知識ベースマシンMu-X(4)ー制御方式の評価ー
伊藤文英、物井秀俊、酒井 浩、仲瀬明彦、武脇敏晃(東芝)
- TM 0549 知識ベースマシンMu-X(5)ー知識ベース演算エンジンー
浅野滋博、柴山茂樹(東芝)、伊藤英則
- TM 0550 論理型言語における多重名前空間の実現について
小長谷明彦、新 淳(日本電気)
- TM 0562 知識ベース管理システムKappaの非正規関係
ユーザインタフェース
小池和弘、宮地泰造、辻秀一(三菱電機)

項インデクスによる知識検索の高速化

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ー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物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演算やプロダクション・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]。

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 $M \times N$ 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のようなになる。先頭の要素が異なる場合には、タイプBの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られるため、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。

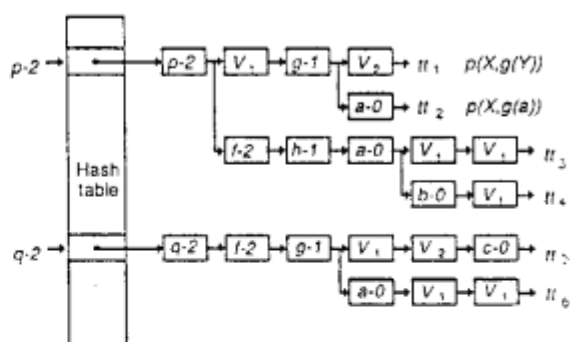


図1. HashとTrieを使ったインデクス

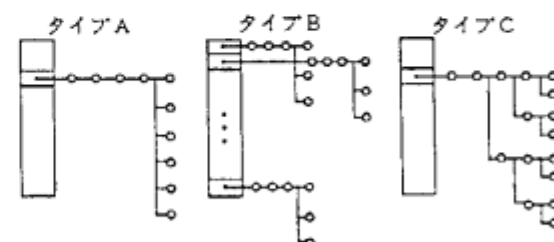


図2. インデクスの形態

4. 評価結果

前述したインデックスの効果を、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のインデックスができるような項関係を生成し、インデックスを張らなかつた場合、インデッ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箇集に対するQuintus-Prologインテ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、インデックスを張ってもHashの効果はないが、Treeの効果によって要素間の比較回数は

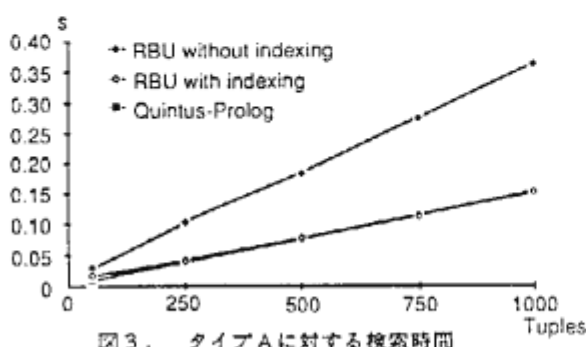


図3. タイプAに対する検索時間

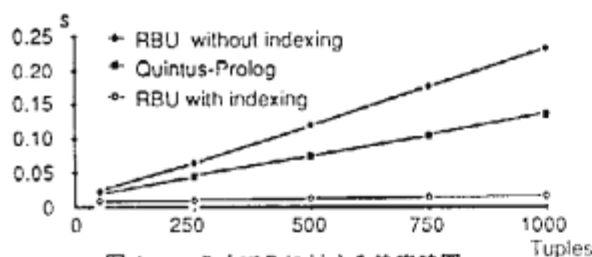


図4. タイプBに対する検索時間

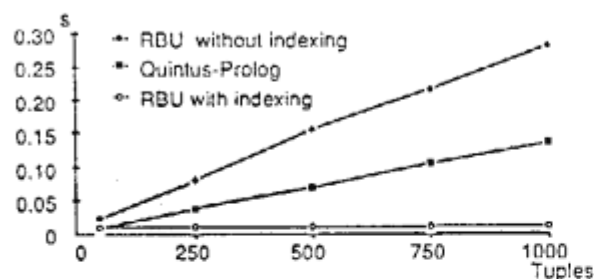


図5. タイプCに対する検索時間

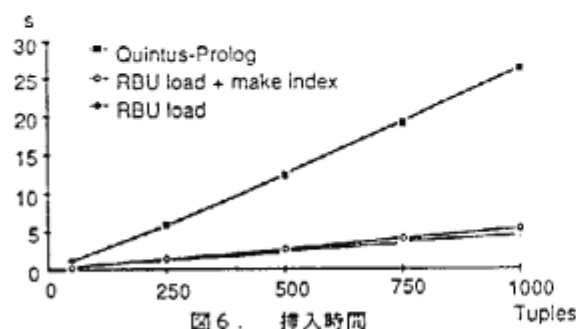


図6. 挿入時間

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、インデッ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のインデッ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のインデッ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。この場合インデックスはTree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ックス維持のオーバーヘッドを評価するための測定も行なった。本インデッ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、削除操作を高速度に行なうことができる。図6にPrologとの挿入速度の比較と、インデックス作成にかかる時間を示した。インデッ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検査器で制御する方法は、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Treeによるインデッ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ー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KB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