MPPMを用いた知識ベースマシン(3)

2C−7　　ー構造体のインデックス方式に関する一考察ー

森田　幸伯、物井　秀俊、　　仲瀬　明彦、柴山　茂樹

(財)新世代コンピュータ技術開発機構、　(株)東芝　総合研究所

1. はじめに

知識情報処理では、基本的なデータ構造として変数を含む構造体が用いられる。推論マシンやエキスパートシステムなど多くの知識情報処理の研究の中でそれら構造体の高速な検索が重要な課題となる。

本研究においても知識ベースの扱うべき基本的な(知識表現)要素として、変数を含む構造体である項を対象としており、その高速な検索方式は重要な課題である。また、本研究では、項を関係の形で管理し、項関係に単一化を用いた演算を定義している。これら演算の高速実行のためにも項に対するインデックス技術が必須である。

筆者らは、重ね合わせ符号(SCW:superimposed code word)を用いた構造体(項)の検索方式を提案した[1]。本稿では、この方式をSSCW (structural SCW)と呼び、他に提案されている方式との関係について述べる。さらに、その関数の構成方式に関する幾つかの性質についてついても述べる。

2. 構造体の検索方式

重ね合わせ符号を利用した構造体の検索方式の研究は、主にProlog処理系の高速化の技法として研究されてきた。ここでいう構造体の検索とは、Prologの項(リテラル)の集合(あるいは、ホーン節集合)に対し、ある項と単一化可能な項の検索を指す。

検索対象の項に変数を含まないと仮定できる場合は、項に含まれる関数記号(アトムも含む)をキーワードとした重ね合わせ符号による部分一致検索を行えばよい(図1-(a))。しかし、検索対象に項に変数を含む場合何等かの拡張が必要である。

[2]では、予め構造を定めその構造内の各位置が変数であるか否かを示すビット列を付加し(図1-(b))、キュアリの項の各関数記号毎に検索対象の項がその関数記号を含むかまたはその位置が変数であるかを調べる方式が提案されている。これによりキュアリの項と単一化可能な項の候補が絞り込める。

その他、各関数記号からのハッシュ値をつなぎ合わせたビット列をインデックス値として利用する方式としてCCW(concatenated code words)[3]やハッシュベクトル[4]やFEW(field encoded words)[5]などがある。そのうちFEWは、重ね合わせ符号のやり方を適用し、変数に対するハッシュ値を、格納用には全て'1'のものを、データ用としては全て'0'のものを用いる方式である(ただし[5]では、格納用には変数の無いものを想定している)。CCWやハッシュベクトルでは、重ね合わせの手法を用いずに変数を特別の値(例えば'0')に対応させて、関数記号のハッシュ値どうしの部分のみを比較している。

[1]では、親ノードからのハッシュ値のレンジを、子ノードのハッシュ値のレンジを覆うように設定する方式(SSCW)を提案している(図1-(e))。この各ノードからのハッシュ値の重ね合わせ方により、従来提案されている幾つかのインデックス方式と等価なものが得られる。例えば、検索対象の項に変数を含まない場合の単純にSCWを用いる方式は、各ノードのハッシュ値のレンジを全て等しくした場合に相当する。FEWは、変数以外のハッシュ関数を子ノードのレンジの部分が'0'になるように制限したものに相当する(図1-(f))。またSSCWでは、変数の位置を指定した検索も、検索用の符号として指定された変数のハッシュ値を全て'1'にすることにより可能となる。

3.SSCW方式のハッシュ関数の構成法

[6]では、確率計算により、各キーワードからの最適なハッシュ関数について議論している。その結果は、重ね合わせられた符号の中で'1'の現れる確率が1/2となるものがDrop率(全体のうち検索されるものの割合)の期待値を最小にするというものである。この節では、変数を考慮した場合(FEWやSSCW)について、同様の議論を試みる。

データ用SCWのiビット目が'1'になる確率をp_iとし、キュアリ用SCWのiビット目が'1'になる確率をq_i、ビット幅をbとする。さらに、計算を簡単にするための近似として、p_iとp_j, q_iとq_j $(i \neq j)$が独立と仮定するとDrop率の期待値は

$$\prod_{i=1}^{b}\left(p_i q_i + (1-q_i)\right)$$

この時iビット目に変数が対応する確率をβ_iとし、α_iを関数記号のハッシュ値の重ね合わせの結果iビット目が'1'となる確率すると、$p_i=\alpha_i+\beta_i$, $q_i=\alpha_i$とかけ、Drop率の期待値は、

$$\prod_{i=1}^{b}\left(\alpha_i^2+(\beta_i-1)\alpha_i+1+\beta_i\right)$$

これは、$\alpha_i=(1-\beta_i)/2$のとき最小で、最小値は、

$$\prod_{i=1}^{b}\left(\frac{3}{4}+\frac{\beta_i}{2}\left(1-\frac{\beta_i}{2}\right)\right)$$

となる。従って、(変数を除いた)各関数記号からのハッシュ関数には、それらを重ね合わせた結果が各

A Knowledge Base Machine with an MPPM (3) - An Indexing Scheme for Terms -

Yukihiro MORITA, Hidetoshi MONOI, (ICOT Research Center)
Akihiko NAKASE, Shigeki SHIBAYAMA. (Toshiba R &D Center)

ビットを$(1-\beta_i)/2$の確率で'1'にするようなものが望ましい。もちろん、$\beta_i=0$のときは、$\alpha_i=1/2$で最小値$(3/4)^b$となり[6]と一致する。

4.簡単な比較実験結果

表1に簡単な実験の結果を示す。表1.ではランダムに生成された100個の項に対する100種の検索の平均Drop率を示している。インデックスのビット幅は、どれも56ビットである。SSCWは、図1-(e)の方式である。ハッシュベクトルは各8ビット7要素である。

表1.ではFEWよりSSCWの方が絞り込みがやや強く、ハッシュベクトルとSSCWは同程度になっている。

FEWやSSCWなどでは、ハッシュ関数としてnビットのうちtビットだけを'1'にする関数が多く用いられる。この場合、区別できる関数記号数は${}_nC_t$(n個の中からt個選ぶ組み合わせの数)個となる。一方、ハッシュベクトルやCCWではnビットに対して2^n個の関数記号を区別できる。従って、この意味ではハッシュベクトルなどの方が分解能が高い。しかし、ハッシュベクトルなどでは、各要素毎に比較を行う為その要素の対応する項内の位置や、そのビット長などは、固定となる。この点FEWやSSCWでは、ビット毎の比較のため各項内の位置に対してビット長を柔軟に設定できる。そのため、表1.では、同じビット幅でも検索される項の数は大差の無いものになっている。

ただし、表1.はひとつのサンプルであり、インデックス方式の性能の評価比較は、検索すべき項集合の性質やキュアリの項の性質などに大きく依存するため、注意深く行う必要がある。また、インデックス方式の比較を行うためにはDrop率だけではなく候補を選び出す手間も考慮する必要がある。比較のための演算については、ハッシュベクトルやCCWよりもSSCWやFEWの方が簡単な演算ですむ。より総合的なインデックス方式の性能評価・比較は今後の課題である。

5.おわりに

重ね合わせ符号を用いた構造体のインデックス方式について、そのハッシュ関数の構成方式と他の方式との関係について述べた。今後このような構造体

表1. 各方式のDrop率

サンプルデータ	単一化可能	FEW	SSCW	ハッシュベクトル
データ1	6.14%	6.56%	6.51%	6.24%
データ2	3.63%	4.00%	3.73%	3.84%
データ3	3.32%	3.98%	3.74%	3.47%
データ4	3.22%	3.72%	3.46%	3.46%
データ5	2.46%	3.11%	2.76%	2.79%
データ6	3.18%	3.75%	3.49%	3.48%

に対するインデックス方式が重要になってくると思われる。知識ベースにおいてもルール検索の高速化や適用することができる。また、単一化結合の並列実行について、項間係の単純な分割では分割損により台数に見合った十分な高速化が望めない[7]。項のインデックス方式を用いることにより組合せの省略が期待できる。また、実際のハッシュ関数決定に際しては、格納すべき構造体の特性により重ね合わ方式やビット幅などを決定する必要がある。具体的なSSCWの構成方式やインデックス方式の単一化結合への応用は今後の課題である。

[参考文献]

[1] 森田他、「スーパーインポーズド方式による構造体の検索方式」第33回情処全大6L-8,1986.

[2] Ramamohanarao, K., and Shepherd J.,"Answering Queries in Deductive Database Systems", *Logic Programming : Proceeding of the Fourth International Conference Vol.2.* ,pp.1014-1033, 1987.

[3] Berra, P.B., et al.,"Computer Architecture for a Surrogate File to a Very Large Data/Knowledge Bases", *IEEE COMPUTER*, March 1987.

[4] 大森他、「推論機能と関係データベースの融合-ハッシュとソートによる検索」第32回情処全大1M-6,1986.

[5] Wise, M. J., and Powers, D. M. W.,"Indexing PROLOG Clauses via Superimposed Code Words and Field Encoded Words", in *Proceedings of the IEEE Conference on Logic Programming*, Atrantic City, NJ, January 1984, pp. 203-210.

[6] Roberts, C. S., "Partial-Match Retrieval via the Method of Superimposed Codes.", in *Proceedings of the IEEE , vol.67. No 12*, Dec."79.

[7] Sakai, H., et al, "A Simulation Study of a Knowledge Base Machine Architecture", *5th International Workshop on Database Machines*, 1987, (to appear).

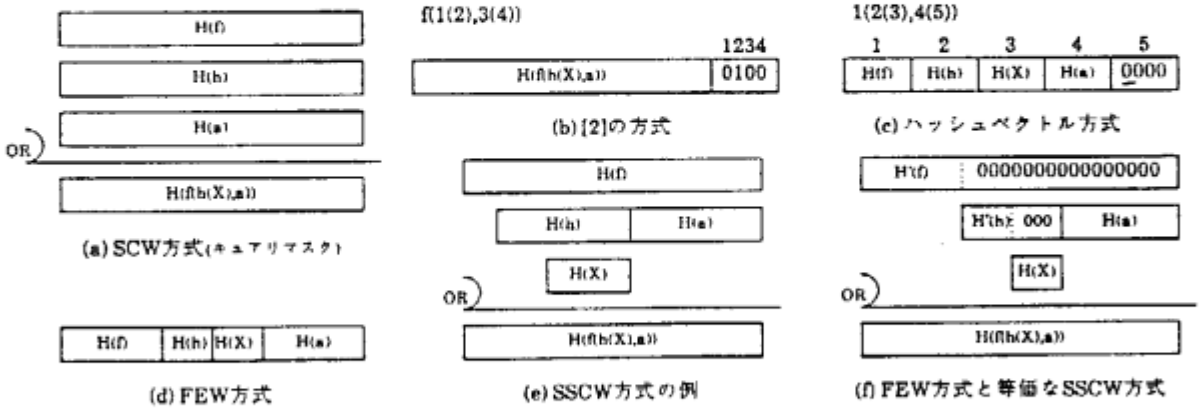

図1. f(h(X),a)の各方式によるインデックスの例