4B-8 連想メモリを用いた単一化のASCA上での実現と評価

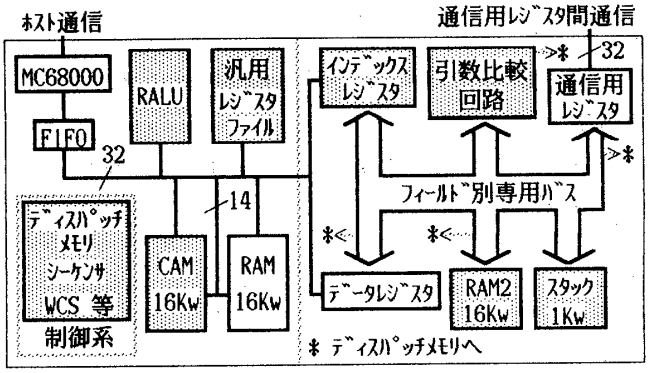
大久保 雅且 (京都大学工学部)
長沼 次郎 (NTT電気通信研究所)

1．はじめに

単一化(unification)は、論理型言語の基本操作であり、高速化は重要である。[1]で示したように、単一化はUNION-FIND問題[2]に帰着でき、連想メモリ(CAM)を利用することによって、高速に処理できる。本稿では、連想メモリを利用したPrologマシンASCA上にこの方法を実現し、その評価を行う。

2．PrologマシンASCA[3]

PrologマシンASCAの単一化部の構成を図1に示す。ASCAは、連想メモリの利用と、多方向条件分岐機構等の専用ハードウェア化を図ったPrologマシンである。また、マイクロプログラム制御により、汎用的なCAMシステムとしても適用できる。ここでは、ASCAのこれらの機能及びCAMを利用し、これを制御するマイクロプログラムを作成することによって、[1]のアルゴリズムをASCA上で実現し、その性能評価を行う。



：今回使用した部分

図1 ASCAの単一化部の構成[3]

3．実現法

単一化すべき項は、項グラフによって表す。項グラフの内部表現(格納形式)は、ASCAのハードウェア構成に合わせた形式となる。内部表現、及びその例を図2～3に示す。ここでは、5ビットのタグを利用し、項グラフの節点及びポインタを表現する。"cタグ"は、各節点からの出力枝の有無の判定に使うタグである。また本方法では、CAMの初期状態をall 0とし、以後の操作でCAMにデータ(束縛節点)を書込んでゆくが、処理途中で、CAMにデータが入っているか、初期状態のままであるかを判定するために"定義タグ"を使う。

入力となる項グラフはRAM2にあらかじめ格納しておき、節点番号はRAM2上での節点のアドレ

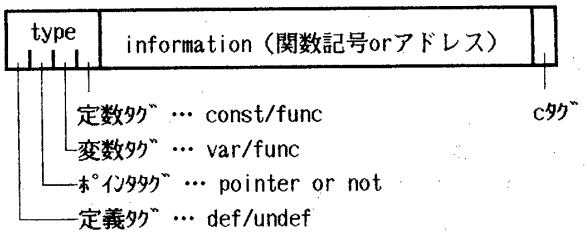


図2 節点のデータ

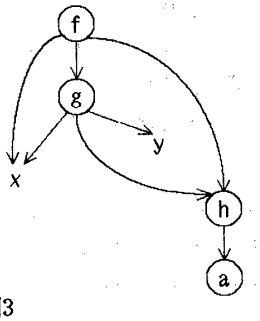


図3
f(x,g(x,h(a),y),h(a)) の項グラフとその内部表現
(太枠部は節点に対応)

addr	type	inf.	c
1	1100	2	0
2	1011	f	1
3	1000	—	1
4	1100	6	1
5	1100	10	0
6	1011	g	1
7	1100	3	1
8	1100	10	1
9	1000	—	0
10	1011	h	1
11	1010	a	0

Implementation and Evaluation of Unification using CAM on ASCA
Masaaki OHKUBO[1], Jiro NAGANUMA[2]
[1]KYOTO University, [2]NTT Electrical Communications Laboratories

スを利用する。[1]のアルゴリズムの、BINDはアドレスによるCAMの読出し、MERGEはCAMの並列検索と並列書込みの機能を利用する。また、条件判断は多方向条件分岐機構及びRALUのフラグ等を、再帰呼出しはスタックを利用し、それぞれ実現する。

4．評価

下記の例題に対する処理時間を図4に示す。

《例題》

$t_1 = f(x_2, x_3, \cdots, x_n, g(y_1, y_1), \cdots, g(y_{n-1}, y_{n-1}), x_n)$

$t_2 = f(g(x_1, x_1), g(x_2, x_2), \cdots, g(x_{n-1}, x_{n-1}), y_2, \cdots, y_n, y_n)$

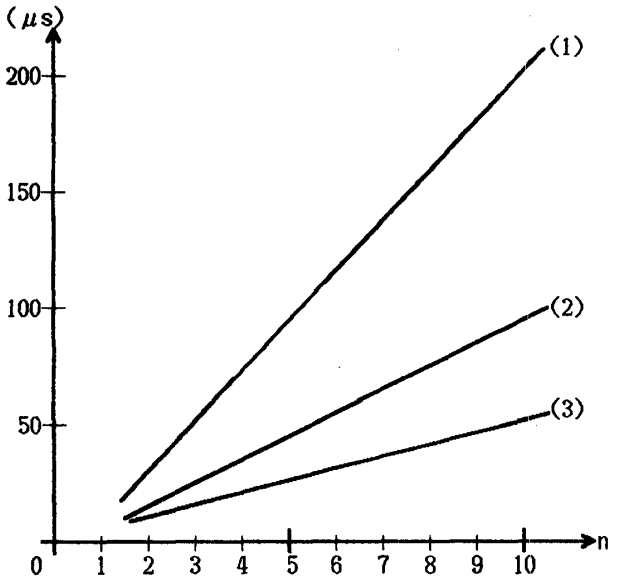


図4 例題に対する処理時間

実際にASCA上で動作させ、実測した処理時間は図4の(1)の直線である。Prolog処理系で利用している単一化アルゴリズムでは、本例題についてはn^2に比例する処理時間を必要とする。これに対し、本アルゴリズムでは直線となることより、その線形性が確認できた。

ASCAは、基本的にPrologの単一化アルゴリズムで動作しているため、バスアーキテクチャや、比較器の位置等が本アルゴリズムに適していない。このため今回は、マイクロプログラムによってアルゴリズムとアーキテクチャの違いを吸収し、実験的に実現した。ここで、アドレスによるCAMアクセスの強化、RAMとCAMの並列アクセスの強化等、本アルゴリズムに適するように若干のハードウェア変更を仮定することにより、処理時間は図4の(2)の直線まで改善できる見通しを得た。

さて、[1]のソフトウェアシミュレーション値を(3)の直線で示す。(2)は(3)の1.5～2倍程度まで接近している。(3)では、3つのモジュールを独立に制御し、データ入力に伴って処理を行うなど、本アルゴリズムに適したハードウェア構成を想定している。これに対し、(2)では、Prologの単一化アルゴリズムを前提としたハードウェア構成となっている。このようなハードウェアの差によって処理時間に差が生じたと考えられるが、(2)のマイクロプログラムの最適化により、両者の差は縮まると予想される。

5．おわりに

連想メモリを利用した単一化アルゴリズムを、PrologマシンASCA上に実現した。これにより、本アルゴリズムの特性及び高速性が確認できた。

Prologは、単一化の他にバックトラックをもう1つの基本操作としている。また、実際のProlog処理系は、メモリ管理法として、構造体複写方式、あるいは構造体共有方式を採用しているが、これらから来る、データ構造や単一化アルゴリズムへの要求も大きい。特に、本アルゴリズムは、単一化すべき項は、項グラフによって明示的に与えられることを仮定している。しかし、実際には、「環境」の下での変数の評価を行わねばならず、これに伴って、項グラフの節点番号とアドレスとの関係が変わって来る。したがって、本アルゴリズムを既存のProlog処理系に適用するためには、何らかのインタフェースを取る必要があろう。このとき、アルゴリズムはどう変わるか、また、その高速性が維持できるかといった点について今後考えたい。

[謝辞]

本研究の機会を与えて頂いた、京都大学工学部矢島脩三教授、ならびに、NTT厚木電気通信研究所集積応用研究室中島孝利室長に感謝いたします。

[参考文献]

[1] 大久保他：連想メモリを利用した高速単一化アルゴリズム，第33回情処全大4B-7

[2] J.E.ホップクロフト,J.D.ウルマン共著；野崎,野下共訳：アルゴリズムの設計と解析Ⅰ，ｻｲｴﾝｽ社,1977，第4章

[3] 長沼他：連想メモリを用いたPrologマシンのハードウェア構成，第31回情処全大2C-1