3E-3

Prolog最適化コンパイラの開発（V）構造体のハッシング方式と動的評価

桐山薫　阿部重夫　黒沢憲一
（日立製作所　日立研究所）

1．はじめに

Warrenによる命令セット[1]をベースにして、コンパイラ方式で高速なプローグ処理系の開発を進めている。文献2）では、Warrenのクローズインデキシング方式を強化するため、最適引数によるインデキシングの機能等を導入したが、本稿では、インデキシングする引数が構造体のときの高速なインデキシングの実現方式と拡張したインデキシング方式の動的な評価結果について述べる。

2．機能拡張したクローズインデキシング方式

ユニフィケーション可能なクローズの検索を高速化するために、Warrenは次の２つのクローズインデキシングの機能を導入している。[1]

・第１引数によるタグインデキシング
・定数、構造体のハッシング

これに対して我々は、次の３つの点について機能の拡張を行なった。[2]

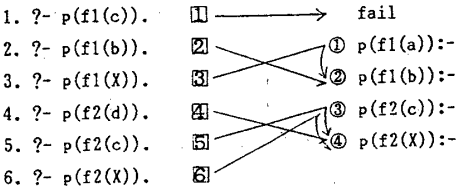
・最適引数によるインデキシング
インデキシングする引数を第１引数に限定せず、タグインデキシングにより別解がなくなる引数、または定数、構造体を多く含む引数をコンパイル時に決める。
・最大２引数によるインデキシング
インデキシングする引数のcaller側が変数のときに、逐次検索となるのを防ぐため、他の引数でもインデキシングできるようにする。
・ハッシングによる構造体の高速なインデキシング
引数の種類が構造体のときは、構造体名と第１要素でハッシングする。

3．構造体のハッシング

3．1　実現方式

構造体名のみでハッシングするときは、同一の構造体名が現われるとハッシングの効率が悪くなる。このため、構造体名の第１要素まで含めてハッシングする方式を考える。これを実現するためには、図１(ａ)のプログラム例から分かるように図１(ｂ)の機能が必要である。ここで、caller側のハッシュ値の求め方を次のようにする。

（１）構造体名の第１要素が定数のときは、構造体名のハッシュ値＋第１要素のハッシュ値との値からハッシュ値を求める。



（ａ）　プログラム例

clause	ゴ　ー　ル	ユニフィケーション
構造体の第１引数は全て定数	1.構造体の第１引数が定数でclause中になし。	fail (1)
	2.構造体の第１引数が定数でclause中にあり。	対応するclauseを選択 (2)
	3.構造体の第１引数が変数	同一構造体名でリニアサーチ (3)
構造体の第１引数に変数を含む	4.構造体の第１引数が定数でclause中になし。	第１要素が変数のクローズを選択 (4)
	5.構造体の第１引数が定数でclause中にあり。	対応するclause及び第１要素が変数のクローズを選択 (5)
	6.構造体の第１引数が変数	同一構造体名でリニアサーチ (5)

（ｂ）　必　要　機　能

図１　第１要素まで含めた構造体のハッシング

（２）構造体名の第１要素が変数のときは、構造体名のハッシュ値からハッシュ値を求める。

そのとき、callee側のクローズからハッシュテーブルを作る方法を次のようにする。

（１）構造体名の第１要素が定数のときは、構造体名のハッシュ値＋第１要素のハッシュ値から求められるハッシュ値、及び構造体名から求められるハッシュ値に対応するハッシュテーブルのエントリに登録する。

（２）構造体名の第１要素が変数のときは、構造体名から求められるハッシュ値に対応するエントリにのみ登録したとすると、ゴールの第１要素がプログラムの中にない定数のとき、ユニフィケーションできなくなる（図１(ｂ)の４参照）ので、ハッシュテーブルの全てのエントリに登録する。

よって、図１(ａ)のプログラムに対するハッシュテーブルは、図２のようになる。この方式では、図１(ａ)1～3の選択効率が悪くなる。そこで、構造体名と第１引数でハッシングするswitch_on_str_arg命令のほかに、

Optimized Prolog Compiler(V)　Structure Hashing
Kaoru Kiriyama, Sigeo Abe, Kenichi Kurosawa,
HITACHI, Ltd.

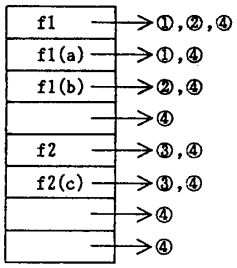


図2　switch_on_str_arg命令のためのハッシュテーブル

表1　実行平均時間比（構造体の第1要素に変数を含むとき）

ρ	1/3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
switch_on_structure命令	1.0	0.93	0.84	0.74	0.63	0.54	0.44	0.35
switch_on_str_arg命令	1.0	0.97	0.92	0.86	0.81	0.76	0.71	0.65

表2　最適引数によるインデキシングの評価結果

評価プログラム	実行時間比(第1引数/第2引数)
8_queen	1.0／0.47
quick_sort	1.0／0.81

表3　switch_on_str_arg命令によるインデキシングの評価結果

インデキシング方法	実行時間比
リニアサーチ／switch_on_str_arg命令	1.0／0.83

構造体名のみでハッシングするswitch_on_structure命令を設け、どちらで実行するかをコンパイル時に決めることとした。

3．2　命令生成基準

クローズ数をm個、構造体名の種類をn1種類、構造体の第1要素の種類をn2種類として、次の2つの場合に分けて考える。

（1）構造体の第1要素に変数を含まない場合

m＝n2，n1＝n2－1とき（構造体名が1ヶ重複しているとき）mが小さければ、switch_on_str_arg命令が高速になる。mが大きくても、switch_on_structure命令とあまり変わらない。そこで、命令生成基準を次のようにする。

$n1 < n2$　のときは switch_on_str_arg命令
$n1 \geq n2$　のときは switch_on_structure命令　　1)

（2）構造体の第1要素に変数を含む場合

変数は最後のクローズに含まれ、ハッシュ値の重複個数を最大3つとし、$\rho = n1/m$ または$n2/m$ としたときの実行時間を近似式から求めた。このときの平均実行時間比を表1に示す。

(i)switch_on_structure命令のρが0.5以上の時は、switch_on_structure命令の実行時間の方が必ず少ないか、n1に対してn2が増加しても2つの命令の実行時間に大きな差がないため、switch_on_structure命令を選ぶ。

(ii)switch_on_structure命令のρが0.5未満の時は、同一のρに対する2つの命令の実行時間比は同程度なので、1)式の命令生成基準を用いる。

変数の数が増えても近似式は変わらず、重複個数が変わっても平均実行時間比は同じ傾向を示す。

4．性能評価

4．1　最適引数によるインデキシングの効果

8_queenプログラム[4] 中の2つの述語(queen1,notmen)とquick_sortプログラム[5] 中の述語(spilit)について、Warren方式により第1引数でインデキシングしたときと最適な引数（第2引数）でインデキシングしたときの実行時間を評価した（文献5）のquick_sortについては、文献4）のquick_sortと第1引数と第2引数が逆になっている）。表2にその結果を示す。表からわかるように、このインデキシングにより8_queenで2．11倍、quick_sortで1．24倍の高速化が実現した。

5．2　switch_on_str_arg命令の効果

コンパイラの中で用いたレジスタ割り当てプログラムについて評価を行なった。このプログラムは、インデキシングする構造体が全て同一構造体名であるため、switch_on_structure命令によるインデキシングは効果がない。このため、タグインデキシングのみを行なったときの実行時間とswitch_on_str_arg命令を行なったときの実行時間を評価した。表3にその結果を示す。表からわかるように、このインデキシングによって1．21倍の高速化が実現した。

5．おわりに

コンパイラによるプロローグ処理系の高速化を図るクローズインデキシング方式について検討し、インデキシングする引き数が構造体のとき、構造体名と構造体の第1要素を参照するswitch_on_str_arg命令を次のように実現した。

・構造体名のハッシュ値と構造体の第1要素のハッシュ値との和を用いてハッシングの高速化を図る。

・構造体名と第1要素に関するハッシュテーブルは1つとする。

この結果、switch_on_str_arg命令によりベンチマークプログラムでリニアサーチに対して1．21倍の高速化を図ることができた。また、最適引数のインデキシングについて評価した結果8_queenプログラムで2．11倍、quick_sortプログラムで1．24倍の高速化を図ることができた。

参考文献

1)E.Tick,D.Warren:Toward a Pipelined Prolog Processor,International Symposium on Logic Programming, Feb.1984

2)情報処理 第32回全国大会 3F1~3F4

3)情報処理 第33回全国大会 3E3~3E5

4)奥乃:記号処理 28-1, Jun.1984

5)W.F.Clockskin C.S.Mellish:Programming in Prolog, Springer-Verlag, 1981