汎用計算機上のＧＨＣコンパイラ

5Q−4

森田正雄*，吉光宏*，太細孝**，上田和紀***

* （株）三菱総合研究所　** 三菱電機（株）　*** ＩＣＯＴ

1．はじめに

我々は汎用計算機上のＧＨＣ処理系を探索し、コンパイル時の最適化により、十分効率よく動作する処理系を開発することができた。ここにそのコンパイル技法について報告する。

なお本処理系はガード部に制限された述語しか書くことができない Flat ＧＨＣをベースとしている。

2．処理方式

ＧＨＣ［Ueda 86］のプログラムは、ゴールに対応する複数の候補節を並列に試み、ある候補節のヘッド・ガード部で呼出し側を具体化する場合に中断があり、そして呼出し側が具体化された場合には再開始処理により中断した処理が再開されるという同期の機構がある。

ＧＨＣの言語仕様に忠実にインプリメントしようとすると、手続き（同じ述語名をヘッドに持つ節の集合）が呼出された際にその手続きがＮ個の節からなっていれば、Ｎ個の節ごとのプロセス（中断・再開始できる単位）を発生させ、それらのプロセスのうち１つの節がコミットした際には他のプロセスを取消してしまうようなＯＲプロセスにより、ヘッド・ガード部を実行するのが自然である。

しかし、実際のＧＨＣのプログラムを見てみると、ヘッド・ガード部実行において単一待合せ（外部プロセスの変数具体化による手続き駆動のきっかけを１つしか待たず、そのプロセスが中断した場合、中断情報を保持し、再開始する場合は中断した所から再開する。）で処理できることが多い［森田 86］。実行時多大なオーバヘッドを伴うプロセスの生成・消去を最適化することができれば、実行効率を飛躍的に上げることができる。しかしこの最適化は、手続きごとに異なるため、ベースの処理方式でなく、コンパイラによってしか行えない。このような観点から、本処理系ではベースの処理方式をシンプルにし、コンパイラが自由に処理方式を決定できるようにしている。例えば、コンパイル時に手続きが単一待合せ機構で処理することが可能とわかれば、コンパイラ側で単一待合せ機構で処理できるようヘッド・ガードの順序を適宜並びかえしてコード出力するようになっている。

3．ガード部の決定木化

ＧＨＣコンパイラの最適化の妙味は手続き単位にヘッドを含むガード部を決定木としてコンパイルしきってしまうことである。ＧＨＣの言語仕様はそのような道具立てを保証してくれている。それは、

① ガード部でのユニフィケーションの方向が決まっていること

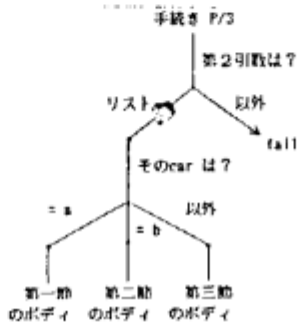
② ガード部で使用できる述語に制限があること

①により中断の起こりうる箇所を静的に全て得られ、またガード部でのシステム述語も中断の原因となりうるが、②によりあらかじめシステム述語の種類及び仕様がわかり、ガード部のシステム述語においても静的に中断や分類のための情報を得ることが可能である。

ガード部の決定木化とは、

```
P(X,[ Y | _ ],Z) :- Y = a | ～ .
P(X,[ Y | _ ],Z) :- Y = b | ～ .
P(X,[ Y | _ ],Z) :- otherwise | ～ .
```

というプログラムを、下図のような決定木に展開して決定的に処理することである。



以下、本コンパイラで行った決定木化の手順を述べる。

手続きの第一引数から見ていき、候補節の集合の中でその引数が不定のままコミットできるものがある場合は、次の引数の検査に移る。そうでないならば主関数記号で分類を行う。分類によっても２つ以上の集合にわけることができず、しかもその引数値が構造体でない場合、次の引数の検査に移る。引数値が構造体ならばさらにその第一要素から再帰的に分類を試みる。この操作を繰り返した後も２個以上の集合に分類できなかった場合は、その引数による分類はあきらめ、次の引数による分類を試みる。また２個以上の集合に分類できた場合には、その引数を決定木のノードとする。そして分類された集合ごとにその要素の数が２以上あれば、その集合について再帰的に分類を試みる。このような手順を繰り返し、最終的な決定木を作る。この手順は、［森田 86］で述べた多重wait checking 技法の拡張となっている。

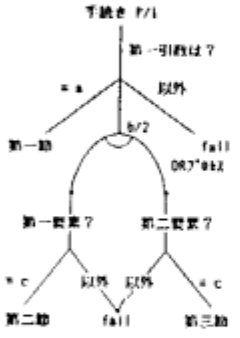
また、決定木の各ノードの処理であるが、この処理（prologのclause indexing に相当する処理）は、現状ハッシュ化による indexing は行わず、逐次に値を比較するコードを出力している。但しその際に、各値ごとにプライオリティが付いており、コンパイル時その順序に従って展開するようにしている。例えばＧＨＣではストリームを扱うことが非常に多い。ストリームは非空リスト構成子と空リストの２種類の関数記号で記述されるが、出現頻度の高い非空リスト構成子に高いプライオリティを与えている。

また手続きによっては、決定木によって節を完全に分類できず、ある候補節に関して多重待合わせを行わなければならない場合がある。例えば、

GHC Compiler on a General-Purpose Computer.
M. MORITA *, H. YOSHIMITSU *, T. DASAI **, K. UEDA***
*MRI, **MELCO, ***ICOT

```
P(a) : - true |     ～ .
P(b(c,_)):- true |  ～ .
P(b(_,c)):- true |  ～ .
```

という手続きは、本コンパイラでは下図のような決定木に展開する。



図中のＯＲプロセスとは２．で述べたＯＲプロセスと同様である。但し効率上の配慮から左図b/2 が選ばれた時点で直ちに２つのプロセスを作るわけではなく、まず第二節を試行し、そこで中断が起こった場合のみ、再開始時中断した所から再開できるよう処理中の状態を保存し、ＯＲプロセスとなる新しいプロセスを作り、そのプロセスで第三節を試行するというように、ＯＲプロセス生成をなるべく遅らせるようにしている。

なお本コンパイラでは、ユニフィケーションやタイプ・チェック、同期及び 'otherwise ' 以外のガード・ゴールは、決定木への展開に利用していない。しかし、決定木への展開にもっとガード・ゴールを利用することも考えられる。例えば、

```
P(X,Y) : - X mod Y = : = 0 |   ～ .
P(X,Y) : - X mod Y =\ = 0 |   ～ .
```

のような手続きでは除算と判定を１回ですませるようなコードを出すことも可能である。このような最適化は今後の課題である。

また本コンパイラでは、決定木化により得られる情報をボディ部の最適化に利用している。例えば、次のようにループ・カウントのためボディ部に算術演算を書くことが多い。

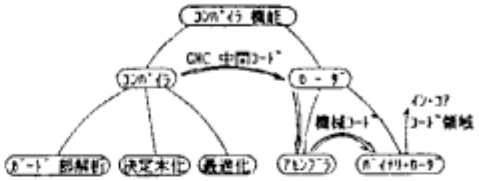
```
P(Counter,Max,--):-Counter =:= Max |   ～ .
P(Counter,Max,--):-otherwise | Ncount:=Counter+1,～
```

このような場合、第二節のボディの算術演算はプロセスとして動作させる必要はなく（変数Counter は先に第一節で中断判定を行っており、第二節を実行する時点では既に具体化されているものとしてよい。）インライン展開し、コミット直後に処理してしまうことが可能である。本コンパイラでは、ボディ部に算術演算が現れた場合に、その評価式中の要素を調べ、その要素が全て中断検査済み変数（決定木化の際の変数情報により判定することができる。）または定数であれば、その算術演算をインライン展開してしまう。

4．　コンパイラ構成

本ＧＨＣ処理系では、コンパイラは、デバッガ等と同様、コマンド（処理系を支援する道具）の１つの機能として提供されている。

コンパイラはイン・コア・コンパイラであり、全てＧＨＣ自身で記述されている。



コンパイラ構成図

（i）　ガード部解析（600 行程度）
　節ごとにヘッド・ガードの解析を行う。

（ii）　決定木化（700 行程度）
　手続き単位に決定木化を行う。

（iii）最適化　（800 行程度）
　最適レジスタの決定及び中間コードの最適化を行う。

（iv）　アセンブラ（800 行程度）
　ＧＨＣ中間コードから機械コードへ落とすプログラムであるが、このプログラムでは、同期のメカニズムによりアセンブル時のラベルが前方参照かそうでないか意識せずにプログラミングできる等、他の手続き型言語等で記述する場合に比べて非常に簡潔に記述することができた。このことは、ＧＨＣのような並列論理型プログラミング言語の言語としての有用性を実際のアプリケーションで示した好例と言える。

5．　評価

本コンパイラのコンパイル時間及び実行性能を下表に示す。

ＶＡＸ－１１／７８０での性能

プログラム	コンパイル時間	実行性能
n-reverse	(2手続き) 740msec	33.2Krps
q-sort	(2手続き)1390msec	19.6Krps
8-queen	(5手続き)3400msec	8.1Krps (注

（注　この測定時GCが１回発生している。なおGCの処理時間を差し引いた性能は10.4Krpsである。

6．終わりに

今後、多くのプログラムをコンパイル・実行し、チューニングしていくことが重要だと考えている。

なお本研究は第五世代プロジェクトの一環としてＩＣＯＴの委託で行ったものである。

<参考文献>

[Ueda 86]　Kazunori, Ueda:
" Guarded Horn Clauses "　LNCS 221, Springer

[森田86] 森田、古光、太細、上田　汎用計算機上のＧＨＣ処理系　第33回情処全国大会 6D-2