1Q-7

ＪＳＩ　ＡＩ　ワークステーション(７)
－Ｐｒｏｌｏｇコンパイラの評価

浅川康夫　田村直之　小松秀昭　黒川利明
日本アイ・ビー・エム株式会社　サイエンス・インスティチュート

１．はじめに

今回、我々が試作したＰｒｏｌｏｇコンパイラ(文献[2]-[5])の性能及び特性を評価する目的で種々のベンチマークテストをおこなったので、その結果を報告する。

今回、我々が試作したコンパイラは、あくまで実験的なものであり、汎用計算機上でいかにして高速なコンパイラが作れるか、また、いったい何処まで高速にできるのかに焦点がおかれた。したがって、実用的な処理系には本来備わっているべき実行時の種々のチェック、ガーベッジ・コレクション、カット、多くの組み込み述語は実現されていない。しかし、基本的な算術述語をサポートするだけでベンチマークテスト([1]など)によるテストがある程度できた。ここで報告するデータはそういう状況での測定結果である。

測定はおもに３０８１Ｋ（ＣＭＳ／ＶＭ）及びＲＴ－ＰＣ上で行われ、いくつかのテストについては３０９０上でも行われた。３０８１Ｋ及び３０９０の実行時間は仮想ＣＰＵ時間で、ＲＴ－ＰＣではシングル・ユーザー時の実応答時間で測られた。

２．ＬＩＰＳ値

現在、Ｐｒｏｌｏｇ処理系の性能を測るの単位としてリストのＡＰＰＥＮＤによるＬＩＰＳ値が一般的である。本コンパイラによって得られたＬＩＰＳ値を表１に示す。

この場合、ヒントとはモード、タイプ、及びｎｏｔｒａｉｌの宣言をあらわし、ヒントありの時には二つのリストを連結することだけに使用可能で、なしの時には通常のＡＰＰＥＮＤとまったく同じに双方向に使うことが出来る。ＡＰＰＥＮＤに関しては、ヒントなしの場合でも十分な性能が達成されたと思われる。さらに、ヒントによって使い方を固定すれば３５％から５５％のスピード・アップが得られた。

３．最適化の効果

表２は３０８１Ｋ及びＲＴ－ＰＣで、ｕｓａｇｅ宣言やｎｏｔｒａｉｌ宣言を全く与えない場合とできる限り与えた場合について、試作したＰｒｏｌｏｇコンパイラのフェイズ２及びＰＬ．８コンパイラの最適化の効果を測定してまとめた結果である。ＰＬ．８コンパイラは、レジスタ間の冗長なデータ移動や冗長な分岐列の最適化などを大域的に行う。前者は、今までＷａｒｒｅｎの抽象マシン命令レベルで考えられていた節レベル最適化に相当する。従って、節レベル最適化によって達しうる性能は表２のヒントなし・ＰＬ．８だけの場合以下と思われ、これは３０８１ＫではおよそＳＯＯＫＬＩＰＳである。これらのデータは、Ｗａｒｒｅｎの抽象マシン命令をそのまま汎用計算機の命令に展開しても十分な性能がでないことを示している。

フェイズ２における最適化はＰＬ．８の最適化と同様に効果的に働いている。両方を組合せた場合の効果が大きいが、これはフェイズ２の最適化が、基本的に、現在のＰＬ．８コンパイラが処理していない部分を対象としている、すなわち二つの最適化処理は質の異なるものであることによる。

４．宣言の効果

表３は新しく導入した宣言の効果を表したものである。これによると、モード宣言だけでは最適化処理にはほとんど役にたたないことがわかる。一方、タイプの宣言はモード宣言だけの場合よりは効果があるものの、それ単独では効果が小さい。モード及びタイプを合わせた宣言(ｕｓａｇｅ宣言)の効果が大きいことがわかる。

モードやタイプが宣言されて最適化処理が十分に行なわれると、相対的にトレイル処理の割合が増える。たとえば、ＲＴ－ＰＣにおいては全体の24%の処理をしめていた。ホスト・クラスの計算機に比べてメモリー・アクセスの遅いワークステーションにおいてはｎｏｔｒａｉｌ宣言によるトレイル処理の除去は効果が大きい。

５．特性

Ｐｒｏｌｏｇ処理系の性能の評価方法としてＡＰＰＥＮＤによるＬＩＰＳ値は、簡単ゆえに広く使われているが、必ずしも処理系の総合的な性能を反映しているとは言えない。現在、他と比較しうるデータとしては文献[1]によるベチマークテストが唯一のもとと思われるので、これにもとづいて特性の評価を行った。

JSI AI Workstation(7) - An Evaluation of the Prololg Compiler
Yasuo Asakawa, Naoyuki Tamura, Hideaki Komatsu, Toshiaki Kurokawa
Science Institute, IBM Japan, Ltd.

その結果良い結果が得られたのは、[1-1][1-2][2-1][4-1][5-1][6-1][7-1][9-1]といったテストであり、逆に悪い結果が得られたのは、[2-2][5-2][11-1][11-2]といったテストであった。それぞれのテストに含まれる処理を考えると、本コンパイラでは、

・定数のユニフィケーション
・決定的な述語呼出し
・リスト処理

などの処理は速く、

・大きなデータ構造の生成
・不特定項のユニフィケーション
・深いバックトラック

などの処理がオーバーヘッドになっていると言える。

データ構造の生成に対するオーバーヘッドは、構造コピー法による実現のためであり、やむをえない。

不特定項のユニフィケーションはインタープリティブな仕事であり、コンパイラによる最適化は期待出来ない。さらに、高級言語をオブジェクト言語としているために、その処理ルーチンへの呼出しが他の処理に比べて重くなっていると思われる。

バックトラックの処理が重くなっている原因も、オブジェクト言語に高級言語を使っていることによると思われる。すなわち、アセンブラでは陽にコードのアドレスを扱うことができるので、チョイス・ポイントを作るには単にコードのアドレスを積み、バックトラックする時には、積まれたアドレスに分岐するだけですむが、高級言語ではそれができないので、計算型ＧＯＴＯなどを使って扱わなければならないためである。

６．終わりに

この研究の目的は、汎用計算機上に高速なＰｒｏｌｏｇコンパイラを実現するための基本的な技術を確立することにあった。 今回、プロトタイプを作成し、評価することによって、我々のアプローチの正しさや可能性が確かめられた。すなわち、

・２レベルの最適化処理を行うことで、汎用計算機上でも高速な処理系が作製可能である。
・タイプやｎｏｔｒａｉｌなどの補助的な宣言が、最適化処理に有効である。
・従来の手続き型言語の最適化技術は、Ｐｒｏｌｏｇにも適用可能である。

さらに、すでに述べたように、Ｐｒｏｌｏｇを高級言語へコンパイルことのメリット・デメリットなども明らかになった。

参考文献

[1]奥野 博、"第3回LISPコンテストと第１回PROLOGコンテストの課題案"、情報処理学会 記号処理研究会資料 28-4、1984

[2]浅川・田村・小松・黒川、"複数のアーキテクチャをターゲットにした高速Ｐｒｏｌｏｇコンパイラ"、情報処理学会 記号処理研究会資料 37-3、1986

[3] Komatsu,H., Tamura,N., Asakawa,Y.,and Kurokawa, T.,"An Optimizing Prolog Compiler",will bepresented at the Logic Programming Conference '86, 1986

[4] Kurokawa,T., Tamura,N., Asakawa,Y.,and Komatsu, H., "A Very Fast Prolog Compiler on Multiple Architectures", will be presented at ACM-IEEE C/S Fall Joint Computer Conference, November 1986

[5] Tamura, N., "Knowledge Based Optimization in Prolog Compiler", will be presented at ACM-IEEE C/S Fall Joint Computer Conference, November 1986

	ＲＴ－ＰＣ	３０８１Ｋ	３０９０
ヒントなし	５６	６１１	１０００
ヒントあり	８７	８２７	１４２０

表 1. ＬＩＰＳ値（１０００要素のリストのＡＰＰＥＮＤで測定、単位ＫＬＩＰＳ）

	3081K		RT-PC	
	ヒントなし	ヒントあり	ヒントなし	ヒントあり
最適化なし	1	1.06	1	1.09
Prologだけ	1.68	1.92	1.62	1.86
PL.8だけ	1.64	1.85	1.68	1.74
PrologとPL.8	3.30	4.47	2.80	4.35

表 2. 最適化の効果（ＬＩＰＳ値の相対値、ＡＰＰＥＮＤの場合）

	3081K		RT-PC	
	notrailなし	notrailあり	notrailなし	notrailあり
宣言なし	1	1.08	1	1.15
モードだけ	1.03	1.26	1.04	1.30
タイプだけ	1.07	1.16	1.23	1.36
モードとタイプ	1.24	1.40	1.47	1.97

表 3. 宣言の効果（ＬＩＰＳ値の相対値、ＡＰＰＥＮＤの場合）