3E-8

論理型言語処理系ＬＯＮＬＩコンパイラにおけるインライン展開方式

竹内　洋一*　加藤　整**　広瀬　正*　中尾　和夫*
*(株)日立製作所　システム開発研究所　**日立ﾏｲｸﾛｺﾝﾋﾟｭｰﾀｴﾝｼﾞﾆｱﾘﾝｸﾞ

1．はじめに

論理型言語処理系「ＬＯＮＬＩ」(Logic Oriented Language Inferencer)は他言語プログラム結合機能等を拡張した述語論理型言語処理系である。ここでは、ＬＯＮＬＩコンパイラにおけるインライン展開方式に関し、算術演算述語を例として取り上げて、その方式と効果について報告する。

2．インライン展開の方針

サブプログラムのインライン展開の直接的な目的は、呼び出し処理にかかるオーバヘッドを除き、実行速度を向上させることであるが、他の最適化処理が大域的に行なえるようになる効果も大きい。これらの効果は、サブプログラムの長さが短い方が、あるいは、呼び出し頻度の高い方が大きくなることが期待される[1]。また、オブジェクトコードの容量の増加は好ましくないが、述語の長さがある程度以下である場合、他の最適化が促進される場合、呼び出し元が一箇所に限られる場合等では増加量が小さいか、減少傾向となる。

以上の点を考慮して、算術演算述語を展開の対象に選んだ。実プログラムにおける調査結果[2]では、組込述語の中では推論制御述語に次いで算術演算述語（この場合には比較演算述語を含む）の呼び出し頻度が高く、多いもので全体の２０～３０％を占めている。

3．展開の方式

3．1　処理の分類

表１に示すように算術演算述語は２引数の演算子から成る述語である。演算子はis、または６種類の等号、不等号である。演算子OP1がisの場合の処理は、式Eの評価値を求め、その値と変数Xとをユニファイすることである。演算子OP1が等号、不等号の場合の処理は、式E1、E2の評価値を求め、それらの値が指定された等号、不等号を満たすかどうか判定し、成功、もしくは失敗終了することである。

式は整数、実数または算術式であり、式の評価値は、整数値、実数値または算術式の値である。算術式は表２に示すように１または２引数の演算子OP2から成る。算術式の評価値は演算子で指定される演算を実行することにより定まる。各引数は式であり、算術式の中に別の算術式が含まれることがあるが、この場合には内側の算術式から順に評価値は定まる。

以上をまとめると、算術演算述語の処理は算術式の評価のための演算子OP2の実行を複数回繰り返し、最後に演算子OP1の処理を１回行うことになる（図１）。

3．2　部分展開の方式

演算子OP1、OP2の実行が算術演算述語の処理の主要部分を成すが、その展開方式を考える上で問題となったのは次の点である。

表１．算術演算述語の分類

引数1	演算子 (OP1)	引数2	
X	is	E	Xは変数
E1	=:=	E2	E,E1,E2は式
	=$=		式は{整数
	<,<=		実数
	>,>=		算術式}

表２．算術式の例

引数1	演算子 (OP2)	引数2	演算の機能
E3	+	E4	加算
	*		乗算
	<<		左シフト
	-	E5	正負反転
	real		実数へ変換

E3,E4,E5は式

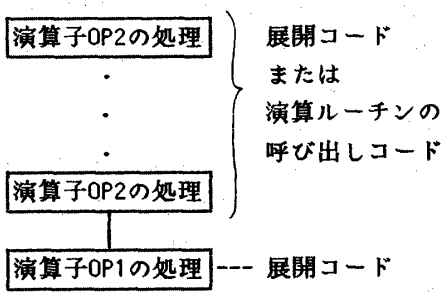


図１．算術演算述語の処理の流れ

Inline Expansion Method for Logic Programming Language LONLI Compiler
Youichi TAKEUCHI[1],Osamu KATO[2],Tadashi HIROSE[1],Kazuo NAKAO[1]
1:HITACHI,Ltd.　2:HITACHI Micro Computer Engineering,Ltd.

(1) 各演算子の引数の型（整数、実数）の判定

(2) コンパイル時に不定な算術式を含む場合の扱い

まず、問題点(1)については、プログラム上から引数の型を自動判定するか、タイプ宣言を利用することにより解決することとした[3]。問題点(2)で想定しているのは、最初、変数であった引数が、評価時までの間に算術式にユニファイされるようなケース、例えば、算術演算述語 Y is Y1 * Y2 （Y1は整数、Y、Y2は変数）において、実行時に Y2 に算術式 Z1 + Z2（Z1、Z2は整数）がユニファイされるケースである。この場合には新たにユニファイされた算術式 Z1 + Z2 に対する処理を事前に展開することはできない。

```
not_take([q|r],q1,q2):-     H
    q =$= q1,               G1
    q =$= q2,               G2
    q3 is q1 + 1,           G3
    q4 is q2 - 1,           G4
    not_take(r,q3,q4).      G5
```

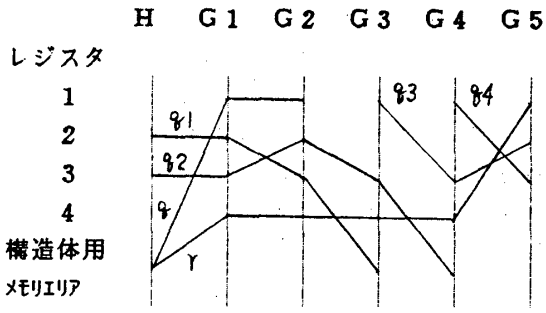
図2．8クイーンプログラムの一節[4]

この点への対策としては、今回、一つの算術式を入力とし、その評価値を出力する演算ルーチンを新たに用意した。その上で、上記のような算術式に対しては展開コードの代わりに本ルーチンの呼び出しコードを生成するようにした。

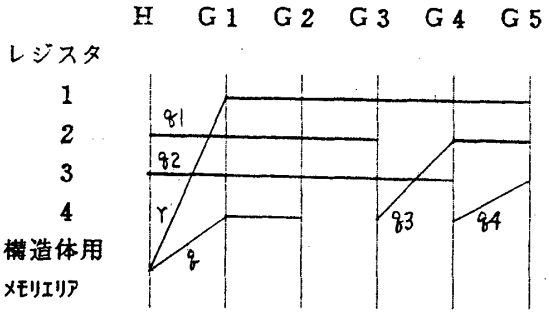
以上が今回の展開方式であるが、本方式の特長は展開できない部分が一部あっても、残りの部分が展開できることである。

4． 他の最適化処理へ及ぼす効果

次に、算術演算述語の展開が他の最適化処理に及ぼす影響について、以下、事例により説明する。図2に8クイーン問題のプログラム内の一節を示す[4]。本節は、ヘッドHと、ゴールG1～G5（G1～G4は算術演算述語）から成る。算術演算述語の展開前後で、本節の実行時の各変数の記憶場所は図3（1）、（2）に示すようになる。展開前では変数のレジスタ間の転送は q1が1回(2-3)、q2が2回(3-2-3)、q3が2回(1-3-2)、q4が1回(1-3)、rが1回(4-1)で計7回である。展開後では、この処理は q3が1回(4-2)、q4が1回(4-3)で計2回となる。このようにレジスタ間の転送の回数が減ったのは、述語の呼び出し回数が減り、変数へのレジスタ割り当ての自由度が増したためである。また、構造体用メモリエリアとレジスタ間の転送は展開前に4回であったのが、展開後には2回に減少している。これは、展開により呼び出し時の構造体の作成が不要となり、レジスタに割り当てることのできる変数が増えたことが原因である。



（1）変数の記憶場所［展開前］



（2）変数の記憶場所［展開後］

図3．変数へのレジスタ割り当ての改善

5． おわりに

今回、3節で示した算術演算述語の部分的なインライン展開方式は、論理型言語の算術演算処理における特性に合った処理の効率化を目指したものである。論理型言語の実プログラムにおける算術演算述語の出現頻度は高いので、本方式の効果は大きいものと考える。

参考文献

[1] R.W.Scheifler:An analysis of inline substitution for a strucured programming language,CACM,Vol.20,No.9(1977)(近山訳;構造的プログラミング言語におけるサブルーチン展開の解析,bit,Vol.10,No8(1978)

[2] 広瀬、他:論理型言語処理系「LONLI」の開発－実用化機能とその効果－、情処学会第31回全国大会5M-4(1985)

[3] 竹内、他:論理型言語処理系「LONLI」における最適コンパイル方式の提案と評価、同第32回全国大会3F-6(1986)

[4] 中島:Prolog、産業図書(1983)