3E-5

Prolog最適化コンパイラの開発（Ⅶ）大域最適化コンパイル方式

阿部重夫　桐山薫　黒沢憲一
（日立製作所　日立研究所）

1．はじめに

Prolog処理系開発の課題である
（1）Prologの高速処理
（2）既存ソフトウェアとのリンケージ
を解決するために、我々は汎用機に内蔵するPrologマシンと、Prolog専用命令に展開する最適化コンパイラの開発を進めている。[1-3]

レジスタ競合を解消してクローズのコードの最適化を図る方式については、既に文献1，2）で報告したが、本論文では、上記方式を拡張してゴールが決定的な組込述語の場合、それを越えて最適化を図る方式について述べる。

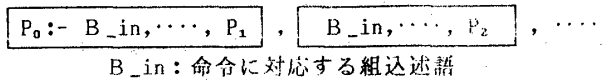
2．最適化方式

2.1　考え方

Warrenの処理方式では[4]、1つのクローズで連続的に実行される単位は、
（1）ヘッドと第1ゴール
（2）第2ゴール以降の各ゴール
となり、コンパイラによる最適化も上記単位で行なわれる。これに対し決定的な組込述語がゴールに現われるとき、レジスタを非破壊にすることができるから、そのゴールを越えた最適化が可能となる。[1-3] このため頻繁に現われる算術演算等の組込述語26種を命令化し、しかも入出力レジスタを任意に指定できるようにした。

このときの最適化は、文献1，2）の方式が容易に拡張でき、図1のヘッドP_0からゴールP_1までの展開を考えると以下のようになる。

Warrenの処理方式では、レジスタを介して引数を引き渡すため、ヘッドP_0から、ゴールP_1までの実行を、ヘッド側のレジスタから、ボディゴールP_1のレジスタへのデータの流れとして定式化しうる。このデータの流れを有向グラフ化し、連結部分グラフにより有向グラフを分割して、連結部分グラフ間の順序関係をレジスタ競合が最小となるように決定する。組込述語命令の実行順序は、その全ての変数を入力側に含む連結部分グラフの後とすればよい。



B_in：命令に対応する組込述語

図1　大域最適化の考え方

2.2　有向グラフ化

Warrenの命令セットでは、リスト・構造体の要素は、一連の命令として展開する必要がある。このため図1のヘッドP_0と、ゴールP_1に現われる変数を次のように分類する。

（1）変数X及びYが同一リスト、構造体中に現われるときは、X，Yは同一の集合Giに含まれる。
（2）変数Xが、X以外の変数が含まれないリスト、構造体中にあるとき、あるいはXがリスト、構造体に含まれないとき、Xは、Xのみを要素とする集合Giに含まれる。

集合Giを用いて連結部分グラフDSGi＝Ii→Oiを次のように定義する。

（1）変数X∈GiがヘッドP₀の第j番目の引数に現われるとき、j∈Iiとする。
（2）Y∈GiがボディゴールP₁の第k番目の引数に現われるとき、k∈Oiとする。

2.3　連結部分グラフの順序付け

連結部分グラフDSGi，DSGj間で、

$$Oj \cap Ii \neq \phi, \quad Oi \cap Ij = \phi$$

が成立すると、DSGiをDSGjより先に実行すればレジスタ競合が発生しない。これをDSGi＞DSGjと表わす。ユニフィケーションに失敗する場合、命令実行のできるだけ早い段階で起きることが必要であるから、連結部分グラフの展開順序を以下のように決める。

（1）Oi＝φとなるDSGi
（2）順序関係のあるDSGi
（3）Ij＝φとなるDSGi

次に組込述語命令の入出力変数の集合をS_Binとす

Optimized Prolog Compiler(VII)　Global Optimizastion
Shigeo ABE, Kaoru KIRIYAMA, Ken'ichi KUROSAWA
HITACHI, Ltd.

ると、

{S_Binに含まれる変数を入力側に含むDSGiで、その変数のユニフィケーションがあるもの}

のうちの最も後のDSGiの後で、組込述語命令を展開すればよい。

2.4 コード生成

DSGiの中には定数データは含まれていないため、これらの展開も含める必要がある。更に変数間のユニフィケーションは、ユニフィケーションの失敗を早めるために先に実行することとすると以下の順序でコードを生成すればよい。

（1）ヘッド中の定数データのユニフィケーション
（2）ヘッド中の同一変数のユニフィケーション
（3）連結部分グラフ及び組込述語命令の展開
（4）ゴール中の定数データのロード

コード生成における最適化の項目は、大きくは以下の2つである。

（1）レジスタベースの命令に展開する。
（2）is, unifyの左辺が変数でかつ、それが初めて現われる変数であれば、これらの命令を省略する。

3. 例題

（例1）次のquick sortのsplitの展開を考える。

```
split([X|L],Y,[X|L1],L2):-
        X =< Y,!,
        split(L,Y,L1,L2).
```

=<，及び！は命令としてあるので、このときのDSGiは次のようになる。（図2参照）

DSG1 = {1, 3}→{1, 3}

DSG2 = {2}→{2}

DSG3 = {4}→{4}

DSG_1～DSG_3は、各々順序関係がないから独立に実行できるが、DSG_2，DSG_3は実行する必要がない。また組込述語命令のS_Bin={X, Y}であるから、DSG_1の後で、X=<Y，！を展開すればよい。従ってこのときのコードは、図3のようになる。

（例2）次のクローズを考える。

```
generate(N,[N|L]):-
        N > 0,
        N1 is N - 1,
        generate(N1,L).
```

このとき

DSG1 = {1,2}→{2}

DSG2 = ∅ →{1}

であり、組込述語の変数の集合 S_Bin={N, N1}であるから、DSG_1の後に組込述語を展開すればよい。組込述語の展開においては、isの左辺のN1は、ここで初めて現われる変数であるから、isは省略できる。このときの命令列は、図4のようになる。

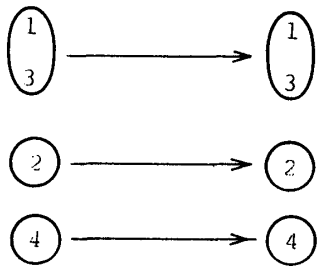


図2 splitのレジスタ競合

```
1.  get_list A1
2.  unify_variable A5
3.  unify_variable A1
4.  get_list A3
5.  unify_value A5
6.  unify_variable A3
7.  less_equal A5, A2
8.  first_cut
9.  execute split, A1
```

図3 splitのコード

```
1.  get_list A2
2.  unify_value A1
3.  unify_variable A2
4.  greater_than A1, 0
5.  decrement A1, A1
6.  execute generate A2
```

図4 generateのコード

4. おわりに

Warrenの処理方式に対し、決定的な組込述語を越えて最適化する方式について述べた。本方式により、レジスタ競合に伴なう命令ステップ数の増加を最小にすることができる。なお本方式の効果については、文献3）に示されている。

5. 参考文献

1．情報処理第32回全国大会 3F1～3F4
2．S. Abe et al, "A New Optimization Technique for a Prolog Compiler", Compcon Spring '86.
3．情報処理第33回全国大会 3E3，3E4
4．D.Warren,"An Abstract Prolog Instruction Set", Technical Note 309, AI Center, SRI, 1983.