3Y-4

Prolog言語処理系ＬＯＮＬＩにおける拡張終端呼び出し最適化方式

広瀬　正*　　古賀　浩二**　　迫田　行介*
*(株)日立製作所システム開発研究所　**(株)日立コントロールシステムズ

1．はじめに

実用システム構築のためのPROLOG言語処理系「LONLI」におけるスタック管理方式(拡張終端呼び出し最適化方式)について述べる。従来のPROLOG処理系で用いられている終端呼び出し最適化方式では、構造データを生成する再帰ループが存在するとグローバルスタック消費が単調に増大し、処理オーバヘッドの大きいガーベージコレクション(GC)処理を行なわねばならない。そのため、実時間性を重視するユーザは、この様な再帰ループが発生しないように配慮してプログラミングする必要があった。ここでの目的は、汎用計算機上での実現に適したスタックを用いながら、その消費量を低減すると共に、消費量を単調増大させないプログラムへの改良を容易化することにある。

2．スタックエリアの回収方針

PROLOG処理系では、実行環境を格納するスタックを、格納するデータの寿命に応じてローカル、グローバル、の2本のスタックに分けて管理する。ローカルスタックには述語呼び出し引き数と実行制御情報(以下フレームと呼ぶ)と節内変数エリアが、グローバルスタックには実行中に生成される構造データを積む。従来の終端呼び出し最適化(LCO)では、ローカルスタックしか回収されない。グローバルスタックを回収するためにはプログラムをバックトラッキング型ループの形式に変更しなければならなかった。LONLIでは、LCOに加えて実行中述語の引き数に着目し、グローバルスタックの回収を図る。

例えば、次のプログラムを考える。

```
..... , p1(A,B),...
........
p1(X,Y):-     ,!,q(Y).
........
q(Z):-        ,!,p2(W), r(V).
```

述語 p2(W) を呼び出す時点(t2)で、以降必要となるデータは X,Y,W,V の値だけである。ここで、グローバルスタックのなかで、述語 p1(X,Y) 呼び出し時点(t1)以降に消費された部分(A(t2-t1)と表す)を考える。このA(t2-t1)部分には、述語 p1(X,Y) 呼び出し時点以降に生成されたデータだけが格納されている。したがって、 X,Y,W,V の値の表現にグローバルスタックのA(t2-t1)部分が使われていなければ、A(t2-t1)の回収、すなわち述語 p(X,Y) 呼び出し時点におけるグローバルスタックの空きの先頭までのスタック回収が可能である。

グローバルスタックの回収を部分エリアA(t2-t1)に対するGC処理で行うという方策も考えうる。しかし、以降必要となるデータ(上例の X,Y,W,V の値)が大きな構造データの場合、それらがA(t2-t1)内に存在しないことの確認、あるいはA(t2-t1)内に存在する部分を抽出と別エリアへの移動のオーバヘッドは大きい。我々は、グローバルスタックの部分エリア内には有効データの値を表現する部分が含まれていないことが容易に確認できる場合だけスタックの回収を行うことにする。この方式ではスタックの回収が行われるケースが限定される。しかし、再帰ループでありながら、スタックが効率良く回収できる機能を持たせておくことにより、プログラミングスタイルの変更を伴わない簡単なプログラムの改良だけで、処理効率の良いプログラム実行が得られる。

3．回収可能条件

次の三つの条件が満たされるとき、グローバルスタックの部分エリアA(t2-t1)は時点t2以降の処理に必要なデータの表現に使用されていないことが保証される。

(条件1) 時点t1において呼び出しす述語 p1(上例の p1(X,Y))が引き数を持たない、あるいは引き数に変数が含まれていない。

(条件2) 時点t2以降に呼びだす述語 p2 (上例では p2(W),r(V))が、引き数を持たない、あるいは引き数に構造データが含まれない。

(条件3) 述語 p1 の呼び出し以降述語 p2 の呼び出しまでの間に非決定性がない。

(条件1)は、述語 p1 はその呼び出し元から(引き数を介して)データを受け取るだけであること、したがって、述語 p1 の処理終了後にA(t2-t1)内に表現されるデータは使用されないことを示す条件である。(条件2)は、述語 p2 の処理に、(引き数を介して)A

An extended last call optimization for Prolog processor LONLI
Tadashi HIROSE*,Kouji KOGA**,Kousuke SAKODA*
*HITACHI, Ltd., **HITACHI Control Systems, Inc.

(t2-t1)内に表現されるデータを渡さないことを示す条件である。(条件3)は、時点t1から時点 t2 までの間の処理にバックトラックがかからないこと、すなわち、時点 t2 以降に述語 p1と p2の引き数データ以外が必要とはならないことを示す条件である。

上述の(条件1)の確認のための処理量は予測しがたい。引き数に大きな構造データが与えられると、そのデータ内部を逐一調べなければならないからである。そこで、もう一歩条件を厳しくした次の条件を用いる。

(条件1)' 時点t1において呼び出しす述語 p1 (上例の p(X,Y))が引き数を持たない、あるいは引き数に変数、あるいは構造データが含まれていない。

さらに、LONLIでは、述語 p2 が規則文の終端項に対応した呼び出しであるときに限り、上述の(条件1)'、(条件2)、(条件3)を判定し、グローバルスタックの回収を行う。この判定機会の限定は、スタック回収の機会を間引くことに相当しスタック消費量の削減効果を減ずるが、この限定により(条件2)の判定対象であるデータが、その時点での呼び出し述語の引き数だけに限定でき、判定処理オーバヘッドは低減できる。

4. 回収アルゴリズム

ここでは、説明簡単化のためにインタプリティブな実行を前提として説明する。処理系は述語呼び出し毎に実行制御情報(フレーム)をスタック(ローカルスタック)上に積みながら処理を進める。処理の成功終了後、失敗終了後に実行すべき述語に対応したフレームをそれぞれ指す前進用ポインタ(FP)、後進用ポインタ(BP)を設けるとともに、これらの値をフレームに退避することにより、フレーム間を前進処理用チェイン(F_chain)と後進処理用チェイン(B_chain)で結合する。LONLIではこれらに加えて、グローバルスタック回収の起点(上述の時点t1に対応)の候補となり得るフレームを指す回収処理用ポインタ(SP)と、これらのフレームを連結するスタック回収処理用チェイン(S_chain)を設けた。これにより、(条件1)'の判定処理の重複を回避し、回収可否判定の効率の向上を図った。図1に、次のプログラムにおける述語 r 呼び出し時点でのフレーム間チェインの状態を示す。

```
u( ) :- ... ,p(xx), ...
p( ) :- ..... ,t( ).
t( ) :- ..... ,q( ).
q( ) :- ..... ,!,s(xx).    (xx部での
s(xx) :- ..... ,!,r( ).     データ授受は
r( ) :- ...                 非構造データのみ)
```

述語 r の呼び出しが(条件2)を満たせば従来のLCOによるローカルスタックの回収のほかに、斜線部分のグローバルスタックも回収できる。

5. 適用例と効果

図2にPrologで記述したあるエキスパートシステムのスタック消費量を示す。(a)、(b)は、それぞれ終端呼び出し最適化を行なわない場合、行なった場合のスタック消費パターンを示している。グローバルスタックにおける消費(上段)が少なくなっている点がここで述べた拡張終端呼び出し最適化の効果である。この最適化導入に伴う実行時間の増大は、インタプタ実行の場合5%以下であった。グローバルスタック消費量が処理の進行と共に単調増加の傾向にあるが、再帰ループ内の非決定性の排除と、任意の終端呼び出しの1つでの構造データの受け渡しを中止することによりこの増加を抑止することができる。

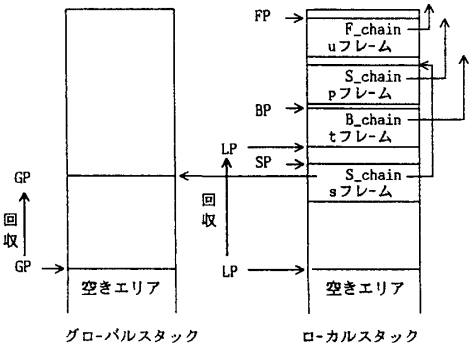

図1. スタック管理方式

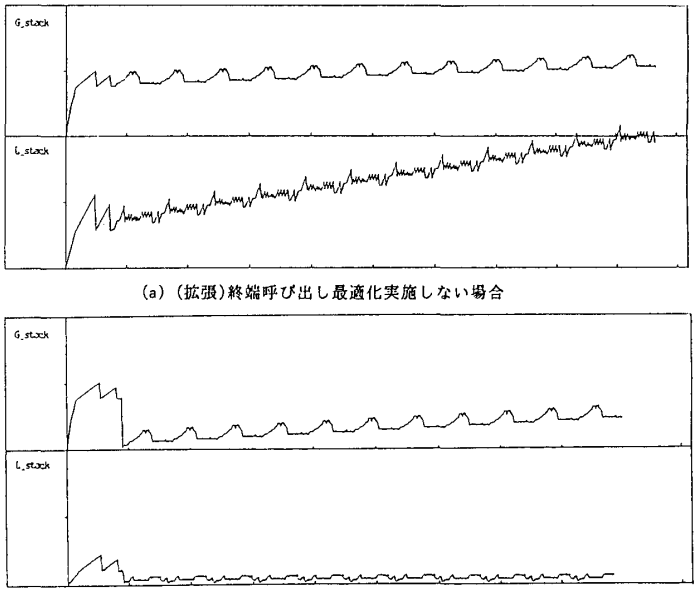

図2. 拡張終端呼び出し最適化の効果