7F−7　関数論理融合型プログラムの高速化手法

中川 康二　中原 鉱一　井田 哲雄
筑波大学

1　はじめに

関数論理融合型言語は、関数型と論理型のプログラミングスタイルを統一的に扱うことのできる言語である。この分野に関しては、様々な研究グループがその計算モデルや実現方法について研究を行なっている。我々のグループでは、関数論理融合型言語の計算モデルとして遅延ナローイング計算系を採用し、その処理系を構築した[1] [2]。しかし、現行の実現手法では、遅延評価とバックトラックが相互に影響し、実行効率が低下することが指摘されている[3]。本稿では、この問題に対する解決法を示す。これにより、飛躍的に効率が向上する。

2　関数論理融合型言語

我々の設計する関数論理融合型言語のプログラムは、等式の集合から成る。例えば、次のようなものである。

```
R={  f(1)=1   f(2)=4   f(3)=9
     g(1,2)=3 g(1,1)=2         }
```

Rのもとで、次の質問の解は次のようになる。

```
? f(3)=X       →  X=9
? f(g(1,1))=X  →  X=4
? f(g(X,2))=9  →  X=1
```

関数だけでなく、最後の例のように論理変数も扱うことできるのが関数論理融合型言語の特徴である。しかも関数評価が遅延評価で行われるため無限のデータ構造を扱うことができるのも大きな利点である。

3　抽象機械

関数論理融合型言語を実行する抽象機械(LNAM: Lazy Narrowing Abstract Machine)として、我々はPrologの抽象機械であるWAMを参考にした。LNAMは、WAMに関数項を扱う機構を拡張したものである。LNAMは、ローカルスタック、ヒープ、トレイルスタック、及びいくつかのレジスタから成る。ローカルスタックは、環境フレームと選択点フレームから成る。選択点フレームは、節の選択のために用いられる。項の表現の最小単位は、セル といい、タグと値の組からなる。節の選択は、選択点(チョイスポイント)を作ることによって実現している。バックトラックが起こると、最新の選択点に従って、選択点作成時の状態に戻される。このとき、トレイルから指されているセルを選択点に記憶されているトレイルポインタまで未束縛にする。

An Optimization method of Functional Logic Languages
Koji Nakagawa , Koichi Nakahara, Tetsuo Ida
University of Tsukuba

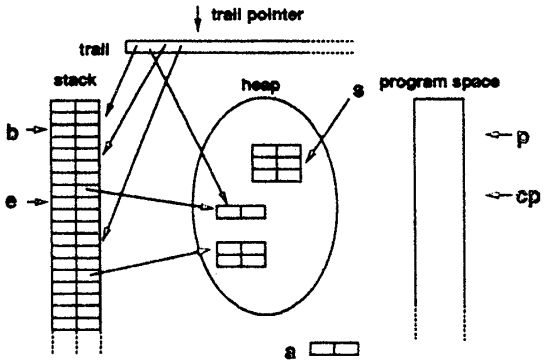

図1．抽象機械 LNAM の構成

4　実現方法とその問題点

関数は、遅延評価によって評価される。この機構のため、関数は関数項として格納されている。関数項の評価結果は、関数項の中のキャッシュフィールドと呼ばれる領域に格納される。このときトレイルスタックからこのキャッシュフィールドにポインタを指しておく。こうすれば、バックトラックの際にこの評価結果も戻されることになる。

しかしこの方法では、関数項が評価される際に、バックトラックによって同様の計算が行われてしまうという問題点がある。このことを例を用いて説明する。

例. 2節で示したRについて、 g(X,Y) という関数項があり、Xが1に束縛され、Yが2に束縛されたとする。するとこの関数項は、g(1,2) となる。このとき、 X,Yのセルは、トレイルスタックから指される(図2(i)(ii))。この関数項がfに渡されたとする。

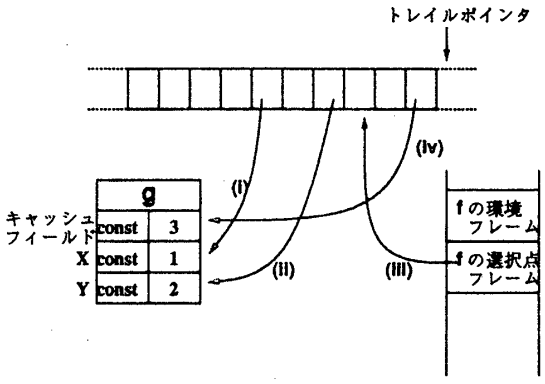

図2．実現方法

スタックには、fの環境と選択点フレームが形成される。選択点フレームには現在のトレイルポインタが格納される (図2(iii))。この後、関数項g(1,2) が評価され3となったとする。このとき、キャッシュフィールドにこの値が格納され、同時にトレイルスタックから指される (図2(vi))。 fの中では、g(1,2) という関数項であるからその値は3である。ここで、バックトラックが起こると関数項のキャッシュフィールドの値が未束縛になってしまう。したがって、このfの関数評価の中で g(1,2) という関数の評価が再度計算されてしまい、実行効率が低下する。

5 高速化手法

関数項の再計算を避けることにより高速化する方法を説明する。評価された関数項に着目すると、バックトラック時、関数項の引数に含まれるデータ構造の一部が未束縛になるとき関数の評価値が変わるので、このとき関数項のキャッシュフィールドを未束縛にする。つまり、あるセルを未束縛にするのに同期して、そのセルを評価に使った関数項のキャッシュフィールドを未束縛にする。この機構を実現するために抽象機械を変更する。

- トレイルの変更
  あるセルに同期して戻されるべき関数項は複数あるので、トレイルの一要素をリストで実現する。その要素はセルへのポインタとこれに同期して戻されるべき関数項のキャッシュフィールドへのポインタからなる。バックトラックの際には、トレイルの要素であるリストから指される要素をすべて未束縛にする。

- 束縛セルの変更
  変数を表すセルが値に束縛されるとトレイルから指されるが、逆にセルからトレイルへのポインタは指されないので、いつそのセルが未束縛になるかを知ることができない。そこで、セルを束縛する時にはトレイルから指されるだけでなく、セルからトレイルにもポインタを張っておく。*

関数項の引数には、束縛されたセルが複数存在する場合もある。このときには、引数の中で一番はじめに未束縛になるセル、いいかえると一番あとに束縛されたセルが未束縛になるのに同期して、関数の評価を戻す。このセルは、関数項の評価に使われたセルのうちトレイルへのポインタの値が一番大きいものである。関数項が評価され、キャッシュフィールドに値を格納する時、トレイルセルにキャッシュフィールドへのポインタを追加する。

例. 先ほどの例で説明する。まず論理変数X, Yにそれぞれ1, 2が束縛される。この時それぞれトレイルから指されるとともに、セルからもトレイルを指しておく (図3(i)(ii))。fの関数呼出しによって環境フレームと選択点フレームが形成され (図3(iii))、この関数項が評価されるとき、その評価結果は3となる。評価される際に用いられたセルはXとYであるが、このうち最も新しく束縛されたセルはYである。これは、XとYがそれぞれが指しているトレイルセルの番地を比較すればわかる。セルYが未束縛になる時この関数の引数が変わりg(1,Y) となる。このときに関数項の評価結果も未束縛にすればよいので、セルYが指すトレイルセルのリストにこの関数項へのポインタを追加する (図3(iv))。ここで、バックトラックが起こっても図3(iii) の位置までトレイルポインタが戻されるだけなので、無駄な再計算は起こらない。さらにバックトラックして、Yが未束縛になれば、リストに繋がれた関数項のキャッシュフィールドへのポインタの先も未束縛にするので、このセルに同期して、関数項のキャッシュフィールドの値が未束縛になる。

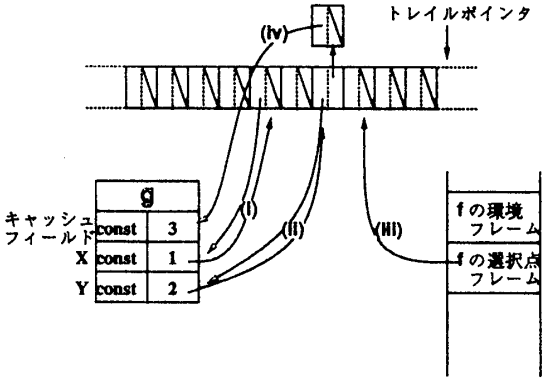

図3. 改良した実現方法

6 おわりに

セルが未束縛になるとき、このセルに同期して、関数項の評価値を未束縛にする機構を加えることによって、無駄な再計算を避ける手法を説明した。これによって、効率が飛躍的に向上する。

今後は、この機構を実装するとともに、関数型言語の実現の効率化手法を導入し、さらに効率化を目指す。

参考文献

[1] 中村, 鈴木, 中川, 井田 : 遅延ナローイング抽象機械のアーキテクチャ. 情報処理学会第44回全国大会予稿,6G-1,1992.

[2] 鈴木, 中川, 井田 : 遅延ナローイング計算系に基づく言語Evとその処理系. 情報処理学会第44回全国大会予稿,5F-7,1992.

[3] Werner Hans, Rita Loogen and Stephan Winkler: On the Interaction of Lazy Evaluation and Backtracking. LNCS 631, 1992.

*実際には、セルにデータとトレイルへのポインタを同時に格納することはできないので、別の方法で実現する。