3G－4

Fleng++ 実験環境

吉田 実[1], 舘村 純一[1], 勝亦 章善[2], 田中 英彦[1]
{minoru, tetamura, katsu, tanaka}@mtl.t.u-tokyo.ac.jp
[1] 東京大学工学部, [2] 富士通*

1 はじめに

コミッティッドチョイス型言語(CCL)は、ストリーム通信を行なうプロセスを利用して、オブジェクト指向プログラミングが可能である。CCLの一つであるFlengを基にしたオブジェクト指向言語Fleng++は、このストリーム通信をするプロセスをオブジェクトとしてプログラマに見せることによって、また、継承機能、インスタンス変数の導入によりプログラムの記述性、可読性を高めたものである[2]。我々は、既にストリーム通信の高速化の手法を提案してきたが、この手法の実装上経験を得、また、ユーザ環境の実現を目的として、Flengインタプリタ上へのインプリメントを行なった。FlengおよびFleng++は一般の並列マシンへのインプリメントが可能であるが、特に我々の研究室で開発中の並列推論エンジンPIE64上の主たるシステム記述およびユーザ言語として採用する予定である。PIE64のハードウェア完成の際、まず、実装の容易なFlengが先に実装される予定である。その際、本実験環境が初版のFleng++処理系として、プログラム環境を提供する。

2 並列オブジェクト指向言語Fleng++の概要

Fleng++は、プログラムのモジュール化の単位としてクラスと呼ばれるものを持っている。クラス定義は、継承クラスの指定、クラス名の宣言、インスタンス変数の宣言、メソッドの定義、ローカル述語の定義からなる。インスタンス変数とは、何度でも値を代入できる変数である。継承には多重継承を採用しており、他のクラスの定義を取り込むことができる。あるオブジェクトは自分や他のオブジェクトにメッセージを送ることによって処理を進める。このメッセージは逐次的に解釈されるが、オブジェクトの内部の実行は並列に行なわれる。

Fleng++は、あるオブジェクトにメッセージを非同期に送ることができるが、メッセージの到着順によって実行結果が変化することがある。特に、インスタンス変数のアクセスにおいては、メッセージの順序性を指定してアクセスしたい場合がある。そのような場合には、メッセージを送る順序を指定できる。これをメッセージの直列化と呼ぶ。Fleng++では以下の規則を設けた。

- 自分自身へのメッセージは他のオブジェクトからのメッセージより優先される。
- 同じオブジェクトへのメッセージの送信は、メソッド定義の中で左から右へ深さ優先で行なわれる。

2.1 オブジェクトの実行

Fleng++のインスタンスオブジェクトは、メッセージプール、ゴールプール、その他の属性からなる。オブジェクトの実行は、メッセージのリダクションとゴールのリダクションに分かれる。メッセージのリダクションは、順次、先頭のメッセージを読み込み、それをリダクションする。一方、ゴールのリダクションは、任意のゴールを並列に取りだし実行することができる。ゴールのリダクションは、Flengのゴールリダクションと同じように行なうことができる。

3 プログラミングの実験環境

3.1 処理系の構成

処理系の全体構成は、図1のようになっている。

Fleng++のコードは、一旦、Fleng gという中間コードに落される。その後に、FlengやCにコンパイルされ、実行時処理系に読み込まれる。

Flenggは、Flengにかなり近いものである。主とした相違は

1. クラス階層を設けた。
2. リスナの定義節が存在する。また、その定義節では、ボディ部に書かれたシステム述語は、左から順に実行される。
3. インスタンス変数、メソッドの記述ができる。

である。1でクラス階層があるので、同じ名前の述語でもクラスが異なれば、別の述語として扱われる。2の規則は、アトミックに行なわれるべき、リスナリダクションにおける効率の重視と、コンパイルを容易にするためである。3は、Flenggのレベルでは、Fleng++の性質をそのまま見せているためである。

3.2 Flengの機能拡張

実行時の環境やPIE64[3]のハードウェアの完成時の移植性の良さを考えると処理系をFlengで書くことが望ましい。よって、Fleng++ → Flenggのコンパイラ、Fleng g → Flengのコンパイラ、実行時処理系をFlengで記述した。そのため、現在、使用可能なFlengインタプリタ上でのスムーズな開発を行なうことができた。現在、実行時処理系を含め1000行あまりである。コンパイラおよび実行時処理系を記述するにあったって、Flengインタプリタの機能拡張を行なった。以下はその内容である。

*"Fleng++ Experimental Environment", YOSHIDA Minoru[1], TATEMURA Jun'ichi[1], KATSUMATA Akiyoshi[2] and TANAKA Hidehiko[1], [1] Univ. of Tokyo, Faculty of Engeneering, [2] FUJITSU Ltd.

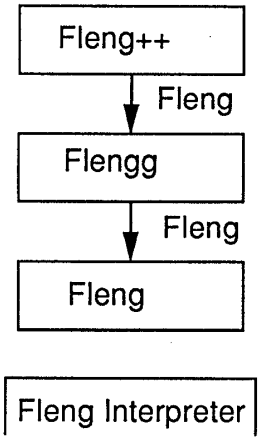

図 1: 処理系の構成

内容のコピー dup

未定義変数を含むタームを読み込むなど、ある値が変数か否かがわからない場合がある。そのような場合に内容をコピーする dup を使用する。また、多重参照の可能性のあるデータをコピーすることにより、単一参照性を保証したり、単一参照の構造体を維持するためにも使用する。

破壊代入 vector_assign

単一参照の構造体は、効率を考えると破壊代入による再利用が有効である。また、キャッシュなどのようにデータをアクセスする時期によって内容が異なっても構わないようなものは、破壊代入が利用できる。本実験環境では、[1] に基づきオブジェクト操作で破壊代入を用いている。

実行中のコンサルト機能

コミッティッドチョイス言語では、通常、assert, retract のような機能は持たない。しかし、コンサルトの機能が実行中に使えないと実行時処理系の中からコンパイルした結果のプログラムを読み込むことができない。よって、Fleng インタプリタに実行中にコンサルトできる機能をつけた。

上記のシステム述語の拡張およびそれを用いたライブラリの拡張によりコンパイラと実行時処理系の記述が可能になった。

3.3 コンパイラ

本論文で報告するコンパイラは 2 つある。Fleng++ のプログラムを Flengg のプログラムに変換するものと、Flengg のプログラムを Fleng に変換するものである。そのコンパイルの例を以下に示す。

Fleng++ のプログラム例

```
class sample.
:a(Self) :- :b(Self), :c(Self).
:b(Self) :- io:write(1,R1).
:c(Self) :- true.
l(X) :- l1(X),l2(X).
end.
```

述語名の先頭に ":" が付くのは、メソッド呼び出しである。また、述語名の間に ":" が入るのは、他クラスのローカル述語呼び出しである。何も付かないのは、ローカル述語である。これをコンパイルすると以下のように展開される。

Flengg へのコンパイル結果

```
:- class(sample).
: a(Self) :- send_msg_to_top(Self,c(Self),true) ,
    send_msg_to_top(Self,b(Self),true).
: b(Self) :- io : write(1,_ab258).
: c(Self) :- true.
l(_ac340) :- l1(_ac340) , l2(_ac340).
:- var([0 , nil]).
:- method([c / 1,b / 1,a / 1]).
```

":-" で始まるのは、宣言文で class はクラスの、var はインスタンス変数、method はそのクラスで定義したメソッドの宣言である。

Fleng へのコンパイル結果

```
sample___a(_ae0bc) :-
    send_msg_to_top(_ada20,c(_ada20),true),
    send_msg_to_top(_ada20,b(_ada20),true).
sample___b(_af444) :-
    io__write(1,_afa58).
sample___c(_b09c8) :-
    {sample__true}.
sample__l(_b1ecc) :-
    sample__l1(_b2814),
    sample__l2(_b3150).
sample____new(! sample,! [0,nil],
    ! [[c / 1,sample___c],
        [b / 1,sample___b],
        [a / 1,sample___a]],
    ! nil).
```

ローカル述語は "クラス名" + "__" + "述語名"、メソッドは "クラス名" + "___" + "メソッド名"、インスタンスオブジェクトを動的に生成するときに使う特殊な述語 new は "クラス名" + "__new" という述語名をそれぞれ使っている。

4 まとめ

Fleng++ でプログラミングを行なうための環境の最低限必要な部分は実現できた。しかし、Fleng++ レベルでのライブラリ、ディバッガなどの整備は、これからの課題である。

なお、本研究は文部省特別推進研究 No.62065002 の一環として行なわれているものである。

参考文献

[1] 吉田実, 田中英彦, "並列オブジェクト指向言語 Fleng++ の優先通信の実装とストリーム操作の高速化", 第 39 回情報処理学会全国大会 2Q4(1989)

[2] 中村宏明, 田中英彦, "並列オブジェクト指向言語 FLENG ++ の実装", 第 38 回情報処理学会全国大会 6Q4(1989)

[3] Hanpei Koike and Hidehiko Tanaka, "Multi-Context Processing and Data Balancing Mechanism of the Parallel Inference Machine PIE64", Proc. of Int. Conf. on FGCS 88', pp.970 - 977.