並列型関数型言語Ｃｍｅｘの開発

1Ｅ−6　　藤本聖　白川洋充　大野豊

(立命館大学　理工学部)

1. はじめに

関数型言語は記述力が高く、代入などの副作用がないため並列処理に向いている。また関数型言語は関数をファーストクラスとして扱うので分散環境での評価も容易に行なえ、疎結合型並列計算機における関数型言語の適用に関する研究がいくつか行われている。[1][2]

我々は、分散ＯＳ上で効率的に評価される並列型関数型言語Ｃｍｅｘ(Concurrent Manipulation of EXpressions) の開発を行っている。Ｃｍｅｘは Miranda[3] をベース言語としている。Miranda は、D. A. Turner によって開発された関数型言語で、各種の関数型言語において提案された機能の集大成とも言うべきシステムであり、次に述べるような特徴を持つ。

・ノンストリクト性
・遅延評価による高階関数や無限リストの実現
・ガード式とパターンマッチングによる簡潔な記述法
・ドット式とＺＦ式を用いたリストの簡便な取り扱い
・型推論機構による強力な型付け

Ｃｍｅｘは、Miranda にプロセスの概念を取り入れることにより並列性をサポートしている。そしてプロセス間通信を用いた複数のプロセスの協調動作による並列処理を実現している。本稿ではＣｍｅｘの基本機構について述べる。

2. 基本機構

2.1 preset句

Ｃｍｅｘにおいて関数は常に遅延評価される。しかし並列処理に於て遅延評価では困った場合が生じる。これはプロセス間通信の場合、転送されるデータが評価済みの値であるか未評価の式であるかは転送するデータ量に決定的な差を生じるからである。従ってプロセス間通信の場合には引数を先行評価する必要があり、このためにpreset句を導入した。Ｃｍｅｘでは Miranda の文法をそのまま使用することができる。それにＣｍｅｘ独自の機能を追加し、これらの機能を全てこの preset句内で実現している。ユーザはこの preset句内でポートの宣言やプロセスの生成等を行う。

preset句は where句と同様に用いる。preset句はその中に複数の preset式を持ち、関数式の評価に先立って全ての preset式が順に評価される。

```
Identifier = Expression          || preset式
```

2.2 プロセス

プロセスは並列処理の単位であり評価の過程で動的に生成される。プロセスは preset句内の preset式にプロセス生成子 {p} を付加することによって生成され、与えられた式の評価を行う。評価の結果、正規形が得られたならばプロセスは処理を終了する。

```
|| プロセスの生成
{p} :: * -> *                    || type
Identifier = {p} Expression  || description
```

上式が評価されるとき、新たにプロセスが生成され、与えられた Expression の評価を行う。生成されたプロセスは Identifier に処理結果を返す。結果が返って来る前に Identifier の値を用いようとしたプロセスは値が返るまで封鎖される。

並列処理においては同じ様な処理を行う複数のプロセスを生成しなければならない事が多いが、Ｃｍｅｘでは array文を用いてそれを簡単に行うことができるようになっている。(図1)

Development of a Parallel Functional Language Cmex
Satoshi FUJIMOTO, Hiromitsu SHIRAKAWA and Yutaka OHNO, Ritsumeikan University

```
fib n = f!n
        preset
        array f (m,0,n)
          = {p} 1                    , m < 2
          = {p} f!(m-1) + f!(m-2), otherwise
```

図1　arrayを用いた関数の定義

2.3　ポートとプロセス間通信

ポートはプロセス間の通信の仲介を行う。送信側のプロセスはポートに値を出力、受信側のプロセスは同じポートから値を取り出すことにより通信が行われる。この場合のデータの送受信は同期方式をとっている。すなわち送信側と受信側が共に揃ったとき初めてデータの受け渡しが行われる。どちらか一方が揃わないなら、もう一方は封鎖される。

ポートは宣言子 useport によって宣言される。ポートの型は port * であり、* はそのポートに対して入出力される値の型を示す。一つのポートは一つの型の値しか扱うことはできない。ポートは受け取った値をリストの形で保持する。ポートに値を出力することはそのリストに値を追加することであり。ポートから値を取り出すことはポートの受け取った値をリストとして得ることである。ポートへの入出力操作を行うために次の二つの命令が用意されている。

```
|| ポートに値を出力する
{>} :: port * -> * -> *        || type
{> Portname} Value             || description

|| ポートから値を得る
{<} :: port * -> [*]           || type
{< Portname}                   || description
```

{>}は Value のコピーを指定されたポートに出力し Value をそのまま返す。{<} は指定されたポートの値のリストを返す。

ポートは、他の関数に引数として渡すことができる。したがって、一つのポートが入出力とも複数のプロセスによって共有され得る。そのため、ポートを用いた通信は多対多の通信となり得る。複数のプロセスが同じポートに値を出力した場合、ポートは先着順に値を保持していく。また複数のプロセスが同じポートから値を得ようとした場合、全てのプロセスはそのポートの値のリストを共有する。

3.　おわりに

並列型関数型言語Ｃｍｅｘの基本機構について述べた。Ｃｍｅｘの適用例として巡回セールスマン問題と食卓の哲学者の問題について考察を行い、この様な問題がＣｍｅｘのプロセス間通信を使ったプロセスの協調動作で記述できることを確認した。

Ｃｍｅｘは現在 Sun 3 上でシステムの構築を行っている。プロセスは SunOS 4.0 の LWP (LightWeight Processes)[4] で実現し、プロセス間通信の低レベルルーチンとして同様に LWP のライブラリを用いている。

参考文献

[1] M. C. J. D. van Eekelen, M. J. Plasmeijer and J. E. W. Smetsers, Communicating Functional Processes, Technical Report 89-3, University of Nijmegen

[2] F. W. Burton, Functional Programming for Concurrent and Distributed Computing, The Computer Journal, Vol.30, No.5, pp.437-450, 1987

[3] D. A. Turner, Miranda : A Non-Strict Functional Language with Polymorphic Types, International Conference on Functional Programming Language and Computer Architecture, LNCS, Vol.201, pp.1-16, Springer-Verlag, 1985

[4] System Services Overview, Sun Microsystems Inc.