5Y-1

# Concurrent Common LISP

阪口哲男　杉本重雄　田畑孝一
(図書館情報大学)

## 1. はじめに

Concurrent LISP(以下CLと略す)はLISPに基づいて開発した協同処理記述用言語である[1][2]。CLはLISP1.5に準拠しており、変数の有効範囲等が異なるCommon LISP[3]の仕様とは整合がとれない。我々はCommon LISPの仕様に準拠し、かつCLのプロセス定義に準拠した仕様を満たす環境管理方式を提案し、新しい協同処理記述用言語Concurrent Common LISP(以下CCLと略す)を設計した。

## 2. Concurrent LISP

CLはLISP1.5に基づいた協同処理記述用言語であり、以下のような特徴を備えている。

1) LISP言語本来の特徴をそのまま備える。
2) LISPに適合した機構の多重プロセス・システムを持つ。
3) 少数の基本並行処理記述関数によって柔軟に並行処理を記述できる。

また、CLではプロセスは

「プロセスはそれだけで完備した能力で形式(form)を評価する実体である。」

と定義し、以下のような特徴を持つ。

1) プロセスは動的に起動され、親子関係を持つ。
2) プロセスはそれぞれに固有の識別子、または親子関係に基づく相対的関係によって識別できる。
3) プロセス間コミュニケーションのために共有変数とメッセージ送受機能を持つ。

3)の共有変数については、ある兄弟プロセス間の共有変数はその親プロセスによって束縛された変数である(図1参照)。

## 3. Concurrent Common LISPの言語仕様

CCLは逐次処理に関してCommon LISPの仕様に準拠し、かつCLのプロセス定義に準拠した並列処理の仕様を満たす。

・プロセスとその環境

CCLにおけるプロセスの定義はCLのものを継承する。CCLにおけるプロセスはCLと同様に共有変数とメッセージ送受機能の2種類のプロセス間コミュニケーション手段を持つが、CLにおける共有変数を介したコミュニケーションはCommon LISPではlexical scopeが採用されているため適用できない。

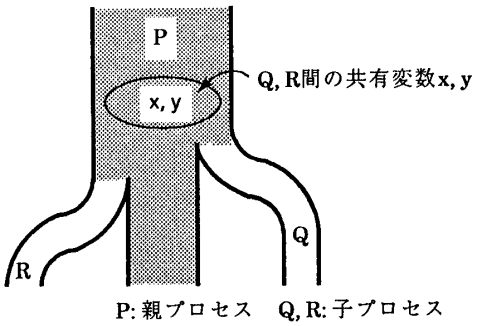


図1　CLにおける共有変数

CCLにおいては、ある兄弟プロセス間の共有変数はその親プロセスの大域変数を用いて表す(図2参照)。一般的には2つのプロセス間の共有変数は共通の祖先プロセスの大域変数を用いて表し、この場合には、大域変数にそれが属するプロセスの識別子を付加する。

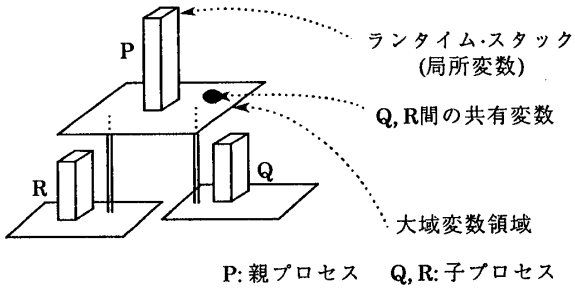


図2　プロセスの階層と変数領域

・データ型と逐次処理機能

CCLの持つデータ型についてはCommon LISPがデータオブジェクトに対し用意している型をそのまま継承する。また単一のプロセス内にのみ限られた処理は逐次に行われ、逐次処理の範囲内ではCommon LISPと同じ環境を提供する。

・並行処理機能

CCLでは、表1に示すようにプロセスを扱うスペシャル・フォームを3つ導入した。

---

Concurrent Common LISP
Tetsuo SAKAGUCHI, Shigeo SUGIMOTO, Koichi TABATA
University of Library and Information Science

表1　プロセスを扱うスペシャル・フォーム

| |
|---|
| starteval: プロセスの生成・起動<br>(starteval (プロセス名1 形式1) (プロセス名2 形式2) …) = (プロセス名1 プロセス名2 …)<br>与えられたプロセス名を持ち、形式を評価するプロセスを生成・起動する。評価値はプロセス名のリスト |
| cr: critical region<br>(cr 形式) = 形式の評価値<br>与えられた形式を他のプロセスと排他的に評価する。評価値は形式の値 |
| ccr: conditional critical region<br>(ccr 条件 形式) = 形式の評価値<br>条件の値がnilでないとき形式を排他的に評価する。評価値は形式の値 |

・その他の機能

以上のような言語としての基本的な仕様に加え、CCLでは入出力機能を強化し、高度なユーザ・インタフェースを構築する基礎として、並行して操作できるマルチ・ウィンドウ機能を備える。

## 4. Concurrent Common LISPの処理系

次にCCLの処理系の実現方式について述べる。CCLは可搬性などを考慮し、UNIX上でC言語により実現を行う。

・CCLデータ領域

CCLは、LISPデータの格納および各種の制御用データを保持するために表2のようなデータ領域を持つ。また、局所変数はランタイム・スタック上に割り当て、かつ大域変数はハッシュ法でアクセスする。それ故、共有変数を含めた変数へのアクセス効率はdeep bindingを用いていたCLに比べ、良くなっている。

表2　CCLデータ領域

| |
|---|
| セル領域 |
| 大域変数表 |
| プロセス・コントロール・ブロック |
| ランタイム・スタック |
| その他 |

・制御構造

CCLの本体制御構造は大別すると表3のように3つの部分からなる(図3参照)。

表3　CCL本体制御部構成

| 制御部分名 | 制御内容 |
|---|---|
| プロセス管理部 | プロセスの生成、終了、プロセス・スイッチング |
| 解釈部 | 関数の評価 |
| ガーベジ・コレクタ | 使用済みセルの回収 |

・入出力制御

CCLでは、入出力機能の内マルチ・ウィンドウ機能はX-Windowシステムに基づいて実現する。この場合、各ウィンドウに対する操作の実現を単純化するため、ウィンドウを直接操作する部分(ウィンドウ制御部)をUNIX上で別のプロセスとして分離する。ウィンドウ制御部とCCL本体プロセスとのコミュニケーションにはバークレイ版UNIXにおけるプロセス間通信機構を用いる(図3)。

## 5. おわりに

本稿ではCCLの言語仕様およびその実現法について述べた。これまでにCCL処理系の設計を終え、現在UNIXワークステーション上で実現を進めている。

参考文献

[1] Tabata, K.; Sugimoto, S.; Ohno, Y. Concurrent LISP and Its Interpreter, JIP, Vol.4, No.4, p.195-202 (1982)

[2] 田畑孝一, 伊藤潔, 杉本重雄. Concurrent LISP, bit, Vol.16, No.6, p.189-204 (1984)

[3] Steele Jr, G. L. Common LISP. Digital Press, 1984, 465p.

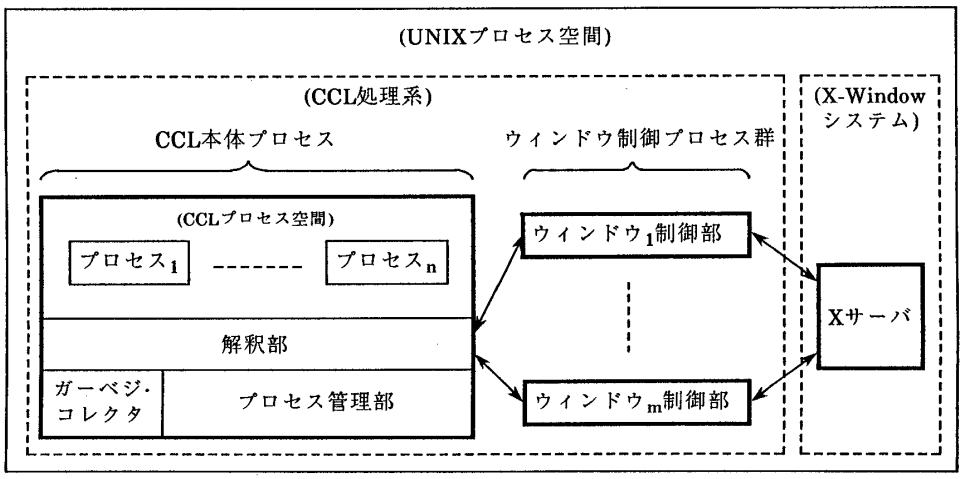


図3　CCL全体構造