5M−7

Commn ESP のマルチプロセス環境

萩原 つね子　内田 洋一　中澤 修

(株) AI言語研究所

1. はじめに

第五世代コンピュータプロジェクトの成果の一つである ESP から生まれた Common ESP (以下 CESP) は，オブジェクト指向機能と論理型機能を融合したプログラミング言語である[1]．論理型言語へのオブジェクト指向によるモジュール化機能導入の試みは CESP 以外の Prolog 系言語でも広く行なわれ，研究目的から商用を目指したものまで数多く存在する[2][3]．

オブジェクト指向融合の形態については言語の記述対象とする分野の違いなどにより種々の方式が考えられるが，それらの多くは CESP と同様まず基本に論理型言語を置き，論理型プログラムを構造化する手法としてオブジェクトを使う．つまり，ホーン節の集合を一つのオブジェクトとして定義する方式で，オブジェクト指向の計算モデルもすべて論理型の枠組に則ることになる．そのため Prolog の実行機構をそのままにしてオブジェクト指向を導入する方式では，メッセージ通信によるオブジェクト同士の並列計算というモデルを表現することができない．

そこで，CESP では言語仕様としてではなく，処理系の持つ環境／ライブラリ機能の一つとしてマルチプロセス実行環境を用意し，並列実行可能な処理のプロセスへの割り付けやプロセス間の通信などを簡単に記述できる枠組を提供することにした．

本稿では，汎用計算機上で実装した CESP 処理系のマルチプロセス機能とその実行環境について説明する．

2. CESP の言語特性

オブジェクト指向の計算モデルとして有効なものの一つに並列に動作するオブジェクトが互いに通信し合って計算するというモデルがある[4]．しかし，このような並列オブジェクトの枠組は Prolog 流のバックトラッキングによる縦型探索機構との整合性は良くない．CESP の言語機能は Prolog の実行機構をベースにオブジェクト指向機能を導入したため，オブジェクト間のメッセージ通信もメッセージの送信側は受信側での処理が終了するまで，次の処理へ移れないようになっている．このような性質はメッセージ通信もすべて Prolog のゴール呼び出しの感覚で使用でき，バックトラッキング制御もすべて統一した枠組で扱えるという利点を持っている．

しかしながら，探索の並列処理を行なうような処理系を作らないことには，並列動作あるいは分散した動作が可能なタスクを記述する場合に現在の CESP の言語仕様の範囲では対応することができない．そこで，CESP 処理系では環境／ライブラリ機能の一部として，一般の汎用計算機上で動作可能なマルチプロセス機能を実装することにした．

3. マルチプロセス機能

(1) 機能概要

CESP は汎用ワークステーションを中心とする各種の計算機上で動作することを目的とする言語である．そこで，CESP のマルチプロセス機能は共有メモリを利用せず，プロセスごとにメモリ空間を独立にする構成とし，CESP 内部からのプロセス生成には OS の持つ fork 機能，通信にはソケットやパイプ機能を利用することにより実現した．

本マルチプロセス機能には

- CESP の子プロセスを生成し，その子プロセス上でサブゴールを実行する
- CESP プロセス同士あるいは CESP プロセスと UNIX 上の任意のプロセス間でサーバ／クライアント型の通信を行なう

の二種類がある．以下それぞれの概要を述べる．

(2) 親子プロセス型

CESP プロセスの実行イメージをそのまま持つような子 CESP プロセスを生成し，その子 CESP プロセスに別の仕事を割り当てるための機能である．

子 CESP プロセスの起動に関しては，親 CESP プロセスとの通信の種類により

a) 同期交信型
　親 CESP プロセスは子 CESP プロセスでの実行終了結果を受け取るまで待ち状態となる
b) 非同期送信型
　親 CESP プロセスは子 CESP プロセスと並行に実行を進める
c) 非同期交信型
　CESP プロセスは子 CESP プロセスと並行に実行を進めるが，いづれ子 CESP プロセスからの実行結果を受け取る

が考えられるが，プリミティブ機能として b) の非同期送信機能と親子間でのデータ送受信機能を用意し，a) と c) はそれらの組合せによるマクロ記法を使用することで実現した．

なお，プロセス間でのデータの送受信機能では相手プロセスに対する標準入出力機能として

　文字，バッファストリング，ターム

を扱うことができる．このとき，入力（受信）側の受信要求はデータが送られない間は待ち状態となる．また，相手プロセスが消滅した場合は受信要求が失敗することになる．

Multi Process Environment of Common ESP
Tsuneko Hagiwara　Youichi Uchida　Osamu Nakazawa
AI Language Research Institute, Ltd.

(3) サーバ／クライアントプロセス型

サーバ機能を持ったプロセスとクライアントプロセスとの通信機能をサポートする．CESP プロセス自身，サーバとなることもクライアントとなることも可能で，CESP 対 CESP 間あるいは CESP 対 UNIX 上の任意プロセス間で通信が行なえる．なお，CESP 対 CESP の通信に関しては，通信用ポート作成，通信および終了などの機能がすべて CESP のメソッド呼び出し（プロセス間通信オブジェクトに対するメッセージ通信）の枠組で利用できる．また，CESP の任意データの送受信を容易に行なうため，通信データの encord/decode 機能も用意した．

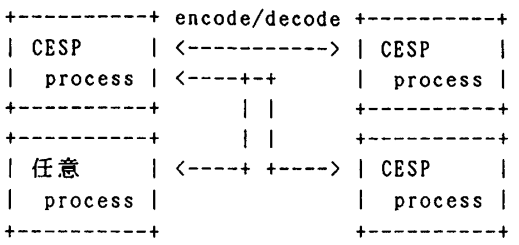

図 1. プロセス間通信の形態

4. マルチプロセス環境

(1) GNU Emacs 上での実行環境

CESP のプロセス環境のベースには GNU Emacs を採用し，必要に応じてプロセスと Emacs バッファを結び付けることができる構成とした．親の CESP プロセスから生成できる子プロセスの機能には

・会話型　　(emacs_process)
・実行専用型 (process)
・トレース型 (process, emacs_process)

の三種類がある．

会話型はクラス emacs_process を使用し，CESP が Emacs 環境上で動作している場合には Emacs の別バッファが生成される．そして新たに生成したプロセスは別バッファからの入出力を受け付けるようになる．プロセス生成用のメソッドは UNIX と同じように fork を使用する．最も簡単なプロセス生成はデバッガ上で

```
?- :fork(#emacs_process, P).
```

と入力することである．この入力により Emacs のバッファは二つに分割され，新たに生成されたバッファでは現在実行している CESP をコピーした子プロセスによってデバッガが起動される．

実行専用型はクラス process を使用する．会話型の emacs_process クラスとの違いは Emacs 環境を使用していてもこのクラスではバッファの生成は行なわずに単一バッファで処理が行なわれる点である．つまり Emacs 環境下でなければ，クラス process と emacs_process の動作は全く同じものとなる．

また，デバッグ時に利用するトレース型は使用しているプロセスクラスに関係なく子プロセスのトレース用に Emacs の別バッファを割り当てる．

(2) プロセスへのサブゴール割付け

(1)で簡単に示したように子プロセスの生成には

```
:fork(プロセスクラス, ^Process)
```

というメソッドを使用する．これは UNIX のシステムコールの fork() と似ており，親プロセスは Process にプロセスクラスのインスタンスオブジェクトを返し，子プロセスにはアトムの child を返す．つまり子プロセス上で「:subgoal(#child)」を実行させるには

```
:fork(#process, Process),
( Process == child, !,
    :subgoal(#child)
;
    親の処理     );
```

というプログラムを書く．この『親の処理』では子プロセスとの通信にプロセスクラスのインスタンスを用いる．例えば親から子へストリングデータを渡すにはこのインスタンスに対して :put1/2 を行ない，子プロセス側ではクラスオブジェクトに対して :get1/2 を行なうことでそのデータを受信できる（図 2 参照）．

```
〈親〉                          〈子〉
process_instance1  <---->  #process
process_instance2  <---->  #process
                    .....
```

図 2. 親子のオブジェクト関係

つまり親は子供の数だけインスタンスを生成し，子供はクラスオブジェクトを使い，一人の親に対して操作を行なうことになる．図 3 に emacs_process の実行例を示す．

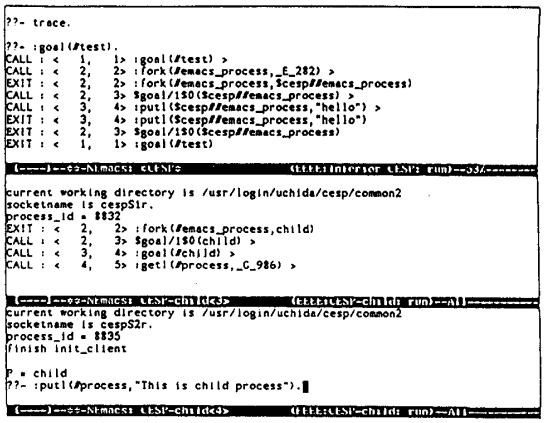

図 3. マルチプロセス環境の例

5. おわりに

本稿では CESP で実装中のマルチプロセス機能／環境について述べた．これらの機能は現在試作を終了し，91 年 4 月配布予定のフルセット仕様版に盛り込まれる予定である．今後は実際的使用環境下での評価を基に改良／拡張を行なう一方，SUN-OS などで実績をあげつつある Lightweight Process の適用についても検討を開始する予定である．

参考文献

[1] 近山，中澤，内田 他：ESPer への道，bit，Vol.22，No.1-12，共立出版 (1988)
[2] LAP USER's MANUAL V3.1 Elas software (1989)
[3] McCabe, F.G : Login and objects, Imperial College Research Report (1986)
[4] 米澤，柴山 他：オブジェクト指向に基づく並列情報処理モデル ABCM/1 とその記述言語 ABCL/1，コンピュータソフトウェア，Vol.3，No.3 (1986)