基本仕様版 Common ESP システムの概要

4G－6

田中吉廣　実近憲昭
(株) AI 言語研究所

1 はじめに

Common ESP(CESP) [1]は、AI言語研究所で研究開発している実用的AIプログラミング言語である。この言語は、論理型とオブジェクト指向型を融合しており、汎用ワークステーション等で稼働する。1988年度は、言語の基本部分を枠組みとしたエミュレータ方式の暫定版CESPを作成した。1989年度は、この暫定版の機能拡張と操作性改善を行ないマシン語生成方式の基本仕様版CESPを完成させようとしている。

本稿では、この基本仕様版CESPの言語仕様と処理系の概要について述べる。

2 言語イメージ

CESPの実行順序の制御は、Prologを基本に、拡張した制御機能を持っており、これに、オブジェクト指向を融合させて、言語体系を形成している。ここでは、CESPの特徴的な機能だけを紹介する。

(1) 論理型の拡張機能

Prologの実行順序の制御機能は、自動バックトラック機能とカットであるが、CESPの場合、これに加えて次の拡張された実行順序制御機能を持つ。

(a) 大域的な脱出制御 (catch/throw/rollback 述語)
多段に呼び出された述語群の中で、深いレベルの述語から浅いレベルの述語へ一度に戻れる大域的な脱出制御を持つ。

(b) 遅延実行制御 (bind_hook 述語)
指定した論理変数に値が束縛された時、指定のゴールが実行される。

(c) バックトラック制御 (on_backtrack 述語)
バックトラック時に、指定のゴールが実行される。

(d) 例外処理の制御 (set_exception_object 述語等)
例外事項が発生した時、指定の例外ハンドラを起動できる。

(2) オブジェクト指向の概念

CESPの基本的制御は論理型であるが、記述力を高めるモジュール化機能として、その論理型プログラムの部分集合をオブジェクトで表現している。また、プログラムの全体の枠組みをオブジェクトの継承／参照で組み立てている。CESPにおけるオブジェクト指向の概念は、次のようなものである。

- クラスオブジェクトとインスタンスオブジェクトを持つ。
- オブジェクト内の述語とスロットはカプセル化できる。
- メッセージ・パシングは、メソッドコールによって行なう。
- 親となるクラスを複数個設定できる多重継承機能を持つ。
- 継承クラス間に分散されたメソッドをある特定の形式に従って結合できる機能としてデモン結合機構とラップ結合機能を持っている。

3 処理系の構成と実行環境

CESPの研究開発マシンは、SUNワークステーションであり、OSは、UNIXである。この上でシステムを開発した後、他の計算機にのせていく予定である。基本仕様版CESPシステムは、高速性能、操作性の良さ、高い移植性をめざしたシステムを構築しようとしている。この実現ため、以下の方式で対応している。

- 高速性能：コンパイラは、最適化したマシン語を生成する。
- 操作性の良さ：操作性のよいEmacsインタフェースを中心に置く。
- 高い移植性：OS依存部の極小化とライブラリ化で対応する。

CESPシステムは、CとCESPで記述している。CESP記述プログラムは、すべてコンパイル後のコンパイルドコードで管理している。このため、最適化したマシン語生成はシステム自体の高速化にも役立っている。

CESPの実行環境を表現すると、図1のようになる。

CESPの実行部は、コンパイルドコードの外部ルーチンからなり、ランタイムルーチン、組込み述語、オブジェクト操作、例外処理、ＧＣ、トレーサ、Ｃインタフェースの必要なコンポーネント群からなる。これらのコンポーネントは、コンパイルドコードおよび例外処理から呼び出され、高速実行するようにＣで記述している。

プログラム管理システムは、プログラミング支援環境である。基本的なコンパイラ、デバッガ、シェル以外にエラー処理を行なうシチュエーションとクラスの検索および操作を行なうライブラリアンからなる。これらは、Emacs環境下で実行できる。

[1]Basic version of Common ESP system
Yoshihiro Tanaka, Noriaki Sanechika
AI Language Research Institute, Ltd.

ライブラリは、大きく分けるとクラス管理、I/O ライブラリ、汎用ライブラリからなる。ライブラリは、ユーザプログラムと同様にコンパイルドコードで管理されている。コンパイルしたプログラムは、CESP が持っているライブラリと一元管理され、統合した形で使用することができる。 これらのライブラリは、CESP システム作成時に選択して保持することができる。

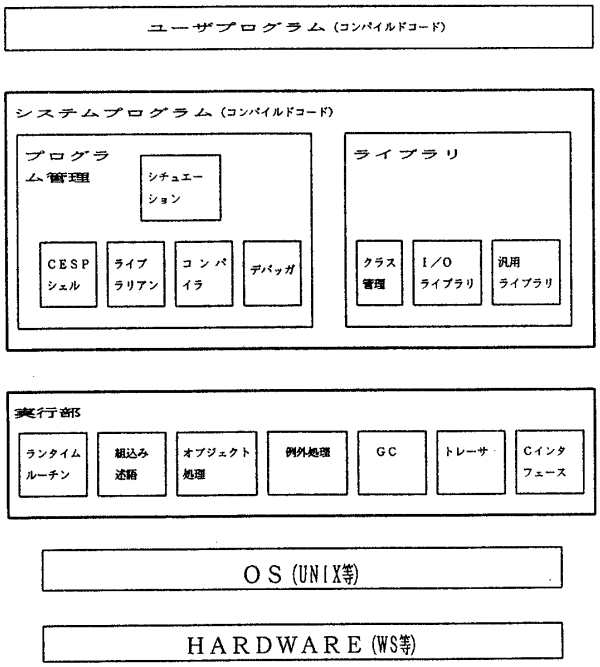

図１　ＣＥＳＰ基本仕様版の実行環境

4　プログラミグ環境

4.1　プログラミング作成支援環境

CESP のプログラミング環境として、Emacs を中心にして、プログラミング環境を構築している。利用者は、ライブラリアン、コンパイラ、Emacs エディタ、デバッガを活用して、CESP のプログラミングができる。 プログラミングの基本的な流れを、次に示す。

(1) CESP+Emacs を立ち上げる。
(2) Emacs エディタにより、CESP ソースプログラムを 作成または修正する。
(3) Emacs の別ウィンドウで、ソースのバッファまたは ファイルをコンパイルする。 コンパイルエラー時は、(2) と (3) を繰り返す。
(4) Emacs の別ウィンドウでデバッガを呼び出し、ソースプログラムを見ながらデバッグする。不良がある時は、(2)(3)(4) を繰り返す。
(5) ユーザプログラムを登録する。

この流れのように、エディタ、コンパイラ、デバッガの繰り返し操作が使い易いように作られている。

CESP は、Emacs の優れたユーザインタフェース機能を活用することにより、優れたプログラム作成支援環境を構築している。

4.2　ライブラリ

CESP は、入出力機能、ウィンドウ操作、プロセス間通信等の OS の依存性が高い機能は、移植性と操作性を考慮してライブラリとして提供している。

I/O に関するものは、現在の研究開発で使用している UNIX での標準入出力、ファイル操作、ディレクトリ操作等を、1 通り揃えている。ウィンドウ操作は、現在、普及しはじめている X-window インタフェースを使用している。 プロセス間通信機能は、UNIX のソケット機能により実現している。これらは、他の OS に移植した時、同等機能が存在する時、システム側のインタフェース部分を変更することにより、移植が容易にできると考えている。

OS に依存しない汎用ライブラリ群としては、プール管理、big num、部分項、文字ユティリティ、シンボライザ、メタ述語、算術関数、コールカウンタ等、豊富なライブラリ群を提供している。

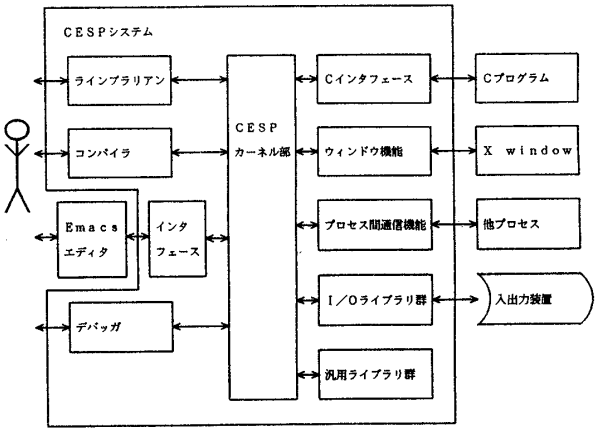

図２　ＣＥＳＰのプログラミング環境

5　まとめ

以上、基本仕様版 CESP の概要について述べた。AI 記述言語の基本的な枠組みは、これで整ったと考える。しかし、研究課題としては、まだまだ検討が必要なことが残っている。

言語仕様では、制約指向との融合やオブジェクト指向の追求があり、処理系では、マルチウィンドウ対応のユーザインタフェースやリアルタイム処理がある。 今後のフルセット版 CESP システムで、このような課題を研究開発する予定である。

6　参考文献

[1] 内田ほか”Common ESP におけるプログラミング環境” 情報処理学会第３９回全国大会 (1989)