Common ESPにおける他言語インタフェース

4G-9

伊藤民哉　中澤修　実近憲昭
(株) AI言語研究所

1. はじめに

第5世代コンピュータプロジェクトにおいて開発された ESP 言語に対して、汎用マシン上で稼働し、さらに機能拡張を行った言語として、我々が開発中の論理型オブジェクト指向言語 Common ESP (以下 CESP) [1] では、プログラムの記述において手続き型言語との組み合せが容易となるように、他言語結合機能が用意されている。

一般的に、論理型言語は、その処理系の構築に当たっては、実行管理やメモリ管理において特殊なメカニズムを持たせるため、通常の手続き型言語とのインタフェースがとりにくくなっている。さらに、CESP においては、オブジェクト指向とのからみもあり、余計に難しくなっている。

本稿では、CESP のオブジェクト指向のスタイルに適合した、他言語インタフェース (以下 FLI) [2] の概要について述べる。

2. FLI の概要

FLI では、以下の機能を提供する。

(1) オブジェクト指向モデルに基づいた、他言語コールスタイル
(2) CESP と他言語間でのデータ受渡し時におけるデータ内部表現の自動変換
(3) 実行に必要な他言語オブジェクトファイルおよびライブラリファイルの自動リンク
(4) CESP のオブジェクトが保有するメソッドを他言語側からコールしたり (コールイン)、また CESP データそのものを他言語側で操作するためのライブラリ関数およびこれらの使用規則

ここで、FLI の簡単なイメージを図1に示す。

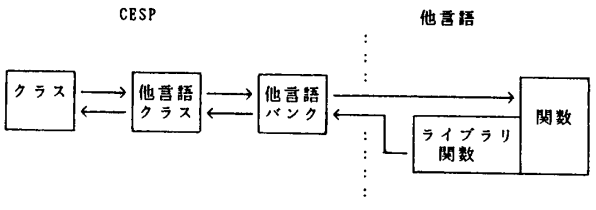

図1 CESP / 他言語 関係イメージ

(4) のコールインに関しては、今年度に開発中の CESP 基本仕様版には、含まれていない。次年度に開発する CESP フルセット版に組み込むものとして、研究中である。

上記 (1),(2),(3) を実現するものとして、以下の定義を導入した。

(1) 他言語インタフェースクラス定義

他言語バンク (後述) を直接使用するクラスを「他言語インタフェースクラス」(以下 FLI クラス) と呼ぶ。CESP のプログラム単位であるクラス定義において、どの他言語バンクを使用するかを宣言することによって、FLI を使用できるようになる。普通の CESP クラスから他言語関数をコールする際は、このFLI クラスへのメソッドコールを行えば良い。FLI クラス定義のシンタックスを図2に示す。

```
〈他言語インタフェースクラス定義〉 ::=
      class〈クラス名〉
      with_foreign〈他言語バンク名〉
      has
        [〈継承クラス宣言〉;]
        [〈クラススロット定義〉;]
        {〈他言語インタフェースクラスメソッド定義〉;}
    [ instance
        {〈インスタンススロット定義〉;}
        {〈他言語インタフェースインスタンスメソッド定義〉;} ]
      end.
```

図2 FLI クラス定義

(2) 他言語バンク定義

CESP から使用する他言語関数の入出力仕様および実行に必要なファイルなどの情報を一括して記述するための枠組みとして、「他言語バンク」と呼ばれる定義を導入した。他言語バンクには、他言語関数コールにおけるデータの入出力関係を主として記述する。これに付随して、他言語バンク内でのみ有効な派生データ型の定義や、別の他言語バンクの継承も行うことができる。他言語バンク定義のシンタックスを図3に示す。

```
〈他言語バンク定義〉 ::=
      foreign_bank〈他言語バンク名〉
      has
        [〈継承他言語バンク宣言〉;]
        [〈派生変数データ型定義〉;]
        [〈記述言語宣言〉;]
        [〈参照オブジェクトファイル宣言〉;]
        [〈参照オブジェクトライブラリファイル宣言〉;]
    [ export
        {〈コールアウトインタフェース定義〉;} ]
      end.
```

図3 他言語バンク定義

Foreign Language Interface of Common ESP
Tamiya Itoh, Osamu Nakazawa, Noriaki Sanechika
AI Language Research Institute, Ltd.

他言語バンク定義は、基本的には他言語オブジェクトファイルと1対1に対応していなければならない。ただし、コールアウトインタフェース定義をもたない他言語バンクでは、参照オブジェクトファイル宣言は必要ない。

他言語バンク定義の中から特に、派生変数データ型定義とコールアウトインタフェース定義について説明する。

(1) 派生変数データ型定義

基本変数データ型および定義済みの派生変数データ型を用いて、新たな派生変数データ型を定義することができる。派生変数データ型定義のために3種類の型構築子を用意している。ここで定義された型は、コールアウトインタフェース定義での引数データ型定義において使用することができる。

(2) コールアウトインタフェース定義

ここでは、CESP と他言語関数間におけるデータ受渡し情報や他言語関数名などのインタフェース定義を記述する。コールアウトインタフェース定義は、他言語インタフェースクラスにおける他言語コール宣言に対応した他言語コールヘッダと、他言語コールボディとから構成される。

実行時には、他言語コールヘッダにおける仮引数の型定義と他言語メソッドコールの実引数とのデータ型が異なる場合には、この他言語メソッドコールが失敗する。また、他言語コールヘッダにおいて、型定義がされていない仮引数に対応した実引数は、他言語側には渡されない（この引数は、他言語コールヘッダのマッチング時にのみ意味をもつ）。

上述の定義を導入して実現されたメカニズムについて簡単に説明する。

（1）データ内部表現の自動変換

他言語関数コールの前後においてデータ内部表現の自動変換を行う。変換の時期を示すために、+,-,= を用いる。これらの意味はそれぞれ、

(1) +
　他言語関数コール前にCESP データを他言語データに変換する。
(2) -
　他言語関数コール後に他言語データを CESP データに変換する。
(3) =
　他言語関数の関数値を CESP データに変換する。

となる。ただし、ある引数に = 指定をした場合は、他言語関数型はその引数の型と同じであるとみなす。また、= は他言語関数コールに付き1つしか指定できず、=を指定した場合には、関数型を指定することはできない。さらに、構造体を受け渡す場合は、構造体渡しではなく、構造体のアドレス渡しとする。これは、実行時オーバヘッドを少なくするために採用している。

（2）オブジェクトファイルの自動リンク

他言語関数の実行に必要なオブジェクトファイルおよびライブラリファイルのリンクは、FLI クラスのメソッドの最初のコール時に自動的に行う。この機能は、他言語バンク内に必要とするファイルを記述することによって使うことができ、この目的のために、link_file および library_file の2つのプリミティブを用意してある。

CESP 基本仕様版では、オブジェクトファイルを直接指定させるが、これを make ファイル指定に換えることによって、もし必要ならば実行時に make を行ってから、他言語関数コールを行うこともできるようになるが、その必要性は現在検討中である。

3．今後の課題

CESP の実行制御メカニズムの中の大域脱出メカニズムおよび、CESP データ領域のガベージコレクションが、CESP と他言語間で多重にコールし合うような場合に問題となることが分かっているので、この解決策を取り込んだ他言語インタフェースを検討中である。この検討を基にした新しい他言語インタフェースを、CESP フルセット版ではサポートしたい。

4．おわりに

以上、CESP 基本仕様版における他言語インタフェースの概要について述べた。CESP 基本仕様版では、手続き型言語で記述された通常の関数（サブルーチンも関数として扱っている）へのインタフェースを優先させ、同時にユーザが使用する際の容易性を高めることを第一の目標とした。このために、論理型処理系における一般的な他言語インタフェースとは異なったものになっていると思う。

今後は、実際的な使用環境下での評価をもとに改良を行い、合わせて CESP の実行制御メカニズムに適合したコールインメカニズムの研究を進めたい。

＜参考文献＞
[1] CESP 言語 （株）AI言語研究所
[2] 中澤他: 論理型言語とオブジェクト指向言語の融合, PROLOG 産業応用シンポジウム論文集 1989