Common ESPにおけるプログラム管理システム

6N-8

佐藤泰典　中澤修　実近憲昭
(株) AI言語研究所

1．はじめに

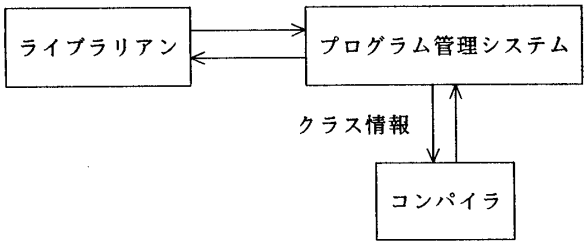
Ｃｏｍｍｏｎ ＥＳＰ（ＣＥＳＰ）は、第五世代コンピュータシステムプロジェクトで開発されたＥＳＰ[1]を基に、さらに高度化、汎用化された言語である。ＣＥＳＰはＰｒｏｌｏｇにオブジェクト指向機能を融合した言語で、ＣＥＳＰシステム自身も基本的にＣＥＳＰで記述されている。本稿では、ＣＥＳＰシステムを構成するサブシステムのひとつであるプログラム管理システムについて述べる。

プログラム管理システムは、プログラムをクラスというモジュール単位で管理している。クラスは大別して、システム提供クラスとユーザ作成クラスに分けられるが、ＣＥＳＰでは同じクラスとして扱い、一元管理を行っている。また、オブジェクト指向言語のプログラミング作成支援環境に必要な機能として、問題解決のための使用クラスの検索、クラス階層構築用の情報の検索等が挙げられている[2]。これらの情報検索機能もプログラム管理システムにより提供されている。

2．プログラム管理システムの概要

プログラム管理システムは、ライブラリアンと呼ばれるユーザインタフェースを介して、クラスの登録、削除、検索情報等の管理を行う。クラスの管理情報は、コンパイル時にコンパイラとクラス情報の交信を行うことにより収集される。

本稿では紙面の都合上、主にプログラム管理システムとコンパイラの間で行われる情報の相互提供、更新等の機能についての説明を行っていく。



図－１　プログラム管理システムの機能構成

3．知的コンパイル機能

オブジェクト指向プログラミングでは継承機能を有効に利用しなければ、コード量の増大や、クラス間のインタフェースの複雑化を招く。ＣＥＳＰでは、プログラミング作成支援環境として、次の様な機能を提供している。

3．1　継承解析コンパイル

実行時のオーバーヘッド軽減のためには、継承関係の解析、メソッドコンビネーションの生成等をコンパイル時に静的に行う必要がある。そのため、継承関係を正しく認識し、継承順に親クラスからコンパイルを行わなくてはならない。多重継承を駆使した大規模なシステムを構築する際には、この継承順コンパイルが非常に煩わしいものとなる。システム側で継承順を自動解析し、適切な順序でコンパイルを行うのが、継承解析コンパイルである。

3．2　差分コンパイル

クラスに変更を加えたとき、その修正が影響を及ぼすクラスを再コンパイルする必要がある。その際に影響のある最小セットのクラスを抽出し、再コンパイルを行うことが望ましい。ＣＥＳＰでは、クラスの最小セットの抽出を自動化し、再コンパイルを行うことができる。

また、各クラスオブジェクトが、自身の生成時間を情報として持っているため、ソースファイルの生成／更新時間と比較することにより、タイムスタンプ機能の利用による最小セットクラスの抽出、および再コンパイルが可能である。

これら２種類の最小セットクラスの抽出機能を利用することにより、大規模システムにおいて非常にプリミティブなクラスを再定義した場合の再コンパイル等においてユーザの負担が軽減される。

3．3　検索機能

アプリケーションプログラムの作成には、システム

Program management system in Common ESP
Yasunori Satoh, Osamu Nakazawa, Noriaki Sanechika
AI Language Research Institute, Ltd.

提供クラスおよびユーザ定義クラスを有効に利用できることが不可欠である。そのためには、ユーザの必要とする機能を持ったクラスおよびメソッドを簡単に検索できなくてはならない。ＣＥＳＰシステムで提供している検索機能では、ユーザの望む機能がどのクラスにあるのか、また既存のクラスの機能とどこが違うのかといったことを効率良くは検索できない。しかし、検索機能を充実させなければ、大規模なアプリケーションプログラム作成の容易性を図ることはできない。現在は、継承／参照クラス情報、スロット情報、メソッド情報、ローカル述語情報等に関する検索機能を提供しているにすぎないが、将来的には、知的なプログラム検索機能を充実させる予定である。

４．コンパイル方式

ＣＥＳＰのコンパイル方式は、４つのフェイズからなる。

(1) 字句／構文解析、マクロ展開、ＷＡＭコード生成：
ソースファイルからプログラムを読み込み、マクロ記述部分を展開し、クラス単位に拡張ＷＡＭコード（クラステンプレート）を生成する。

(2) 継承解析、オブジェクト情報解析、ＷＡＭコード生成：
各クラステンプレートを基に継承解析を行い、オブジェクト指向機能部分を解決した拡張ＷＡＭコード（クラスオブジェクト）と、オブジェクト管理用の各種テーブルを生成する。

(3) 機種独立命令生成、最適化：
拡張ＷＡＭコードから、マシンの機械語に依存しない機種独立な機械語命令を仮想アセンブラの形で生成する。

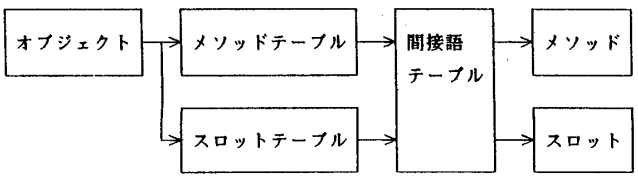
(4) 機種依存命令生成、最適化：
機種独立な仮想アセンブラから、マシン個別の最適化を行った機種依存の機械語を生成する。

実行には (2) と (4) の生成物（オブジェクト管理用テーブルと機械語）を用いる。以下にオブジェクト管理用テーブルの代表的なものを示す。

・メソッドテーブル
メソッド名をキーとしたハッシュ表で、実行時のメソッド探索に用いる。

・スロットテーブル
スロット名をキーとしたハッシュ表で、実行時のスロット探索に用いる。

・間接語テーブル
ＧＣや再コンパイル時のコードやヒープの再配置を簡便にするために用いる。



図－２ オブジェクト管理用テーブルの関係

メソッド呼び出しは実行時に動的に呼び出し先の探索を行う必要があるため、オブジェクト指向機能実現のオーバーヘッドとなっている。一方、デバッグの終了したプログラムは、実行時のオーバーヘッドを極力減らし高速に実行することが重要となる。そのため、ＣＥＳＰではコンパイル時に静的に解決可能なオブジェクト情報等を収集解析し、高速実行用モジュールを生成する機能を提供する。その一方式を以下に示す。

・アドレス固定コンパイル
メソッド呼び出しを間接語テーブルを介することなく直接行おうとするものである。コンパイル時にオブジェクトが特定できるものについては、メソッドの呼び出し先を決定しておき、コードからの直接メソッド呼び出しを可能とすれば、実行時にテーブルを探索することなく高速実行が行える。この直接呼び出しはスロットについても適用可能である。

５．まとめと今後の課題

以上ＣＥＳＰのプログラム管理システムについて述べた。オブジェクト指向プログラミング特有のプログラミング時の負荷を回避するために、プログラム管理システムで各種機能を提供している。今後の課題としては、オブジェクト指向言語としての言語仕様とプログラミング環境を整える上で問題となっている、環境を含めたシステムの選択的な生成やユーザプログラムの実行専用システムの生成がある。また、オブジェクト指向の情報隠蔽機能としてメソッドの可視性制御、オブジェクトのセーブ／ロード等を含めた知識表現に必要な機能として動的オブジェクト[2]の生成等の検討も進めて行く予定である。

参考文献

[1]Chikayama,T.：ESP Reference Manual,ICOT Technical Report,TR-044(1984)

[2]横手：オブジェクト指向言語のプログラミング環境,情報処理,Vol.30,No.4(1989)

[3]尾内 他：動的オブジェクト指向プログラミングの提案と評価,電子情報通信学会論文誌,Vol.J71-D,No.12(1988)