5Q-7

HiOBJ-2における総称関数制御方式

湯浦克彦　(日立製作所　中央研究所)
船津　隆　(日立ソフトウエアエンジニアリング)
堺原　健　(日立製作所　ソフトウエア工場)

1．はじめに

オブジェクト指向言語の性能に目を向けた場合、問題となるのは総称関数が呼び出されてから適当なメソドを起動するまでの処理(以下、総称関数制御と呼ぶ)の効率である。Common Lisp上のオブジェクト指向言語CLOS(Common Lisp Object System)においては、その特徴であるメソド結合を含めた総称関数制御の効率が重要な課題である。本稿では、CLOS準拠のHiOBJ-2[1]における総称関数制御方式について述べる。

2．総称関数制御の概要

2．1　総称関数制御手順

CLOSの各種メソド結合機能を汎用的に処理するための総称関数制御手順を図1に示す。

総称関数が呼び出されると、総称関数の各メソドに対し、与えられた引数が適用可能か調べる。そして、適用可能なメソドが集められると、それらを優先順にソートする。ソートされた適用可能メソドに、メソド結合の方式を適用し、適用可能メソドを呼び出す手続きを表す形式を生成する。この形式を評価することにより結果が得られる。

2．2　総称関数制御の問題点

前述の手順を毎回実行するのは効率が悪い。効率向上のためには、メソド選択とメソド結合の負荷を軽減する必要がある。これまでの効率向上策としては、メソドをキャッシュする方法が多く用いられてきた。この方法はある引数のクラスに対する1回目の総称関数呼び出しにおいて選ばれたメソドの情報をキャッシュしておき、2回目からはキャッシュされた情報を用いることにより、メソド選択の負荷を軽減するというものである。

ところが、この方法をCLOSに適用しようとすると、メソドが追加、削除された場合などに、キャッシュされた情報の更新が容易でないという問題がでてくる。また、依然、メソド結合の負荷の問題が残ってしまう。

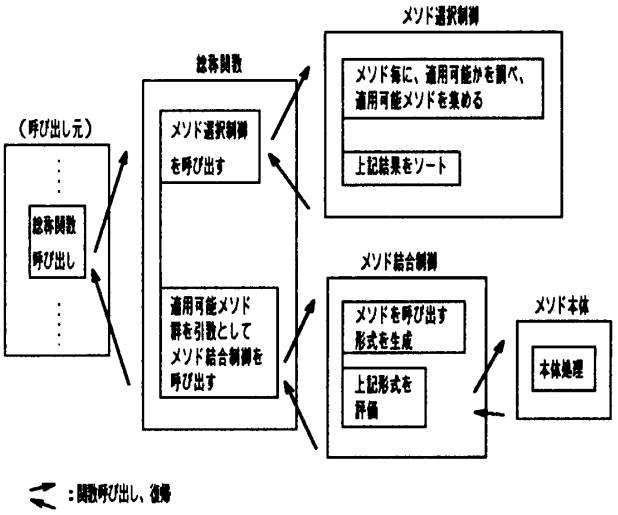

図1　総称関数制御手順

3．メソド結合キャッシュ方式

そこで、HiOBJ-2では前述の問題を解決するために、図2に示すように、メソド結合を適用してメソドを起動する手続きを関数としたメソド結合関数と、適用可能メソド群とをキャッシュし、それらを用いてメソドを起動するメソド結合キャッシュ方式を採用した。

3．1　キャッシュテーブルの構成

キャッシュテーブルは、総称関数により管理され、総称関数毎に1つである。その実体は、キー部が木構造になったリストである(図2参照)。キー部の階層レベルが第1引数からの引数の並びに対応する。登録する情報は、キー部には実引数のクラスオブジェクト、値部にはメソド結合関数と適用可能メソド群である。

3．2　実現方法

(1)　キャッシュテーブルの探索

総称関数が呼び出されると、各必須実引数のクラスをもとにしてキャッシュテーブルを探索する。探索は、第1引数から順に行う。まず、実引数のクラスが、各引数に対応する階層レベルのキー部にあるかを検索する。該当するものがあれば、そのクラスのクラス継承が、登録後も変わっていないかを判定する。ここで、継承変更の判定は、クラスオブジェクトのPrecedence flag[2]で行う。以上の判定を最後の引数まで繰り返す。

途中で、該当するキーがなかった場合には、実引数の型に合うメソドを調べ、キャッシュテーブルに新しいキーと値を追加する。また、継承の変更ありと判定された場合には、新しい継承定義に基づいて、実引数の型に合うメソドを調べ直す。そして、キャッシュテーブルのキーと値を修正する。

このような探索方法により、クラス継承の変更に対応できるとともに、継承変更に伴うキャッシュテーブルの情報の更新にかかる負荷を必要最小限に抑えることができる。

An Implementation of Generic Function for HiOBJ-2 based on CLOS
Katsuhiko YUURA (1) Takashi FUNATSU (2) Ken SAKAIBARA (3)
(1) Central Research Laboratory,Hitachi Ltd.
(2) Hitachi Software Engineering Co.,Ltd.
(3) Software Works,Hitachi Ltd.

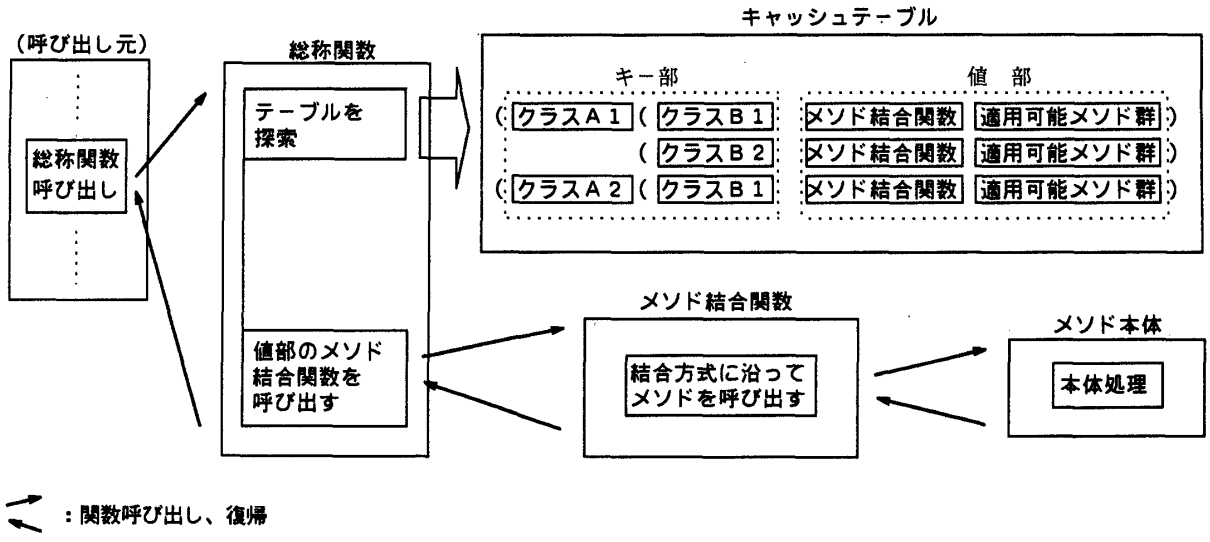

図2　メソド結合キャッシュによる総称関数制御

（2） メソド結合制御

キャッシュテーブルに登録されていることがわかると、該当する値部にあるメソド結合関数を呼び出し、メソド結合を適用して適用可能メソドを起動する。メソド結合関数の引数は、対応する適用可能メソド群である。このメソド結合関数は、システム提供の関数で、表1のように、メソド結合方式と適用可能メソドの種類によって分類されている。それぞれのメソド結合関数の処理は、メソド結合を適用してメソドを起動する手続きを最適化したものである。例えば、表1の第2結合関数は、図3のような処理を行うものである。

以上のような、登録情報の利用とメソド結合制御の特殊化により、メソド選択とメソド結合の負荷を軽減し、総称関数制御を高速化することができる。また、クラスの継承の変更による登録情報の更新にも小さな負荷で対応できる。

3．3　メソド追加、削除への対応

キャッシュテーブルの情報の更新は、クラスの継承の変更の場合だけでなく、メソド追加、削除の場合にもおこる。そこで、メソド追加の場合は、そのメソドが適用可能となるようなケースがキャッシュテーブルにあれば、対応する値部の適用可能メソド群に追加する。メソド削除の場合は、各値部の適用可能メソド群に削除しようとするメソドがあれば削除する。

キャッシュテーブルの値を常に正しいものにするために、メソド追加、削除時にキャッシュテーブルを検索する負荷がかかるものの、情報の更新の負荷は、当該メソドを追加、削除するのみであるため小さい。

4．おわりに

本方式により、次のような効果を得た。

(1) メソド結合を含んだ総称関数制御を高速化することができる。

(2) キャッシュテーブルの登録情報の更新の負荷を軽減することができる。

今後は、HiOBJ-2の総称関数制御の性能を測定し、本方式の有効性を定量的に評価する予定である。

表1　メソド結合関数の分類

メソド結合の種類	適用可能メソドの種類	関数名	メソド結合関数の処理
標準メソド結合	基本メソドのみ	第1結合関数	特定基本メソドを呼び出す。
	基本メソドと :before/:after メソド	第2結合関数	:beforeメソド群、特定基本メソド、:afterメソド群を呼び出す。
	基本メソド、 :aroundメソドと :before/:after メソド	第3結合関数	特定:aroundメソドを呼び出す。:beforeメソド以下の呼び出しは、第2結合関数を呼び出す手続きを介して行う。
組込みメソド結合、短形式宣言的メソド結合	基本メソドのみ	第4結合関数	基本メソドとオペレータを呼び出す。
	基本メソド、 :aroundメソド	第5結合関数	特定:aroundメソドを呼び出す。基本メソドの呼び出しは、第4結合関数を呼び出す手続きを介して行う。

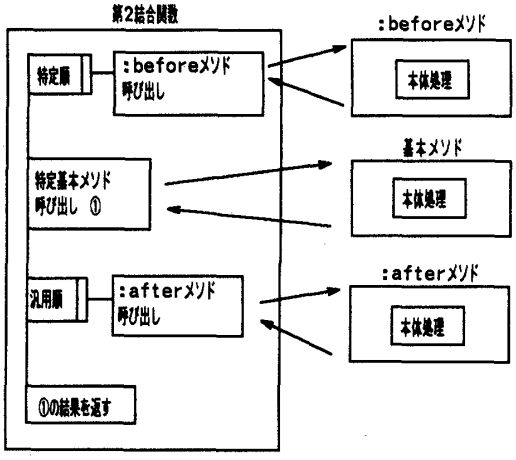

図3　メソド結合関数の処理例

参考文献

1) 湯浦,外：CLOS準拠のHiOBJ-2の概要
情報処理学会第39回全国大会,1989

2) 高橋,外：HiOBJ-2におけるクラス変更方式
情報処理学会第39回全国大会,1989