2M−5

PaiLispのための並列オブジェクト指向言語PaiObject

飯塚 泰樹　伊藤 貴康
(東北大学 工学部 情報工学科)

1. はじめに

並列 Lisp 言語や並列オブジェクト指向言語に関する研究が、最近国内外でさかんである。PaiLisp[1] は Scheme[2] に future, pcall, par, pcond, ... といった並列構文 (Multilisp や Qlisp の主なコンストラクト全てを含む[3]) を導入した共有メモリ型並列 Lisp 言語である。

この PaiLisp をオブジェクト指向言語に拡張した並列言語 PaiObject を設計、試作したので、その概要を報告する。PaiObject は、オブジェクト, クラス, メッセージ通信等のオブジェクト指向を支援する機能を並列プロセス環境下で実現するように設計されているが、プロセスを first-class object として許す点に特徴がある。本論文では、PaiObject の言語構文とその PaiLisp による記述、及び試作した処理系の概要について報告する。

2. PaiObject

2.1 設計方針

PaiLisp は、「PaiLisp = Scheme + 並列構文」と考えられるのに対し、PaiObject は、「PaiObject = PaiLisp + 並列オブジェクト指向」という考えで設計されている。また、並列オブジェクト指向は Oaklisp[4] 構文をもとに、PaiLisp に親和性の高い並列構文を導入するという立場から設計されている。本言語は、プロセスを first-class object として扱っている、多重継承によるコードの共有、一つのオブジェクトへの複数プロセスからの同時アクセスの許可 (すなわち、複数プロセスによるオブジェクトの共有)、明示的なプロセスの生成、排他メソッド、新しいクラスの概念 (すなわち、process,continuation,virtual) 等に特徴がある。また、プロセスを制御するためのメソッド、process-kill,suspend 等が用意されている。

2.2 PaiObject の構文

PaiObject の構文は次のように与えられる。

```
式 E   ::= K | I | (E E*)                                  1
       | ⟨一般の Scheme 式⟩                                 2
       | (spawn E) | (suspend) | (exlambda (I*) E E*)       3
       | (future E) | (pcall E E*) | (par E E*)            4
       | (pcond (E E*)+) | (pcond# (E E*)+)                5
       | (par-and E*) | (par-or E*) | (delay E*)           6
       | (loop E*) | (pmap E E*)                           7
       | (add-method (E (C I*) I*) E E*)                   8
       | (add-exmethod (E (C I*) I*) E E*)                 9
```

また、クラス及び重要なメソッドには次のようなものがある。

```
クラス C   ::= object | class | operation                   10
           | process | continuation | virtual              11
           | ⟨ユーザ定義クラス⟩                              12
           | ⟨その他の Scheme の data-type⟩                  13
メソッド   ::= make | initialize                            14
           | process-kill | process-init | suspend | etc .. 15
```

式 E のうち、1–6 行は PaiLisp のもの、add-exmethod が PaiObject 固有のものになる。また、クラスのうち、process, continuation, virtual、メソッドのうち process-kill, process-init, suspend が PaiObject 固有のものとなっている。

2.3 構文の意味

オブジェクト

PaiObject では、言語中のすべてのものは、オブジェクトとして平等に扱っている。クラスもオブジェクトである。オブジェクトには、(1) インスタンス変数環境、(2) クラスへのポインタ、(3) メソッド起動の機構、(4) 排他機構がそなわっている。

PaiObject は多重継承を許すクラス階層を持つ。すべてのクラスは、class クラスのインスタンスであり、また、クラス階層の最上位には、object クラスがある。

オブジェクトの作成

オブジェクトを作るには、make メソッドが使われる。ユーザクラスを定義するには (make class ..)、インスタンスを生成するには (make C ..) とする。make メソッドは、作ったオブジェクトに対して initialize メッセージを送る。

メソッド

メソッドは、シンボルに対して定義されるのではなく、operation オブジェクトに対して定義される。メソッドは、その定義された環境、オブジェクトのインスタンス変数環境を合わせた環境で実行される。これにより、Scheme のクロージャの意味を壊さないようにしてある。メソッドを定義するには、add-method 式を使う。下の式で、*op* は operation クラスのオブジェクトである。

(add-method (*op* (*C iver-list*) . *arg-list*) *E E**)

排他メソッド

add-exmethod は、排他同期のための排他メソッドを定義するのに用いる。ある一つのオブジェクトへの排他メソッドの適用は、順序付けられて実行される。

メッセージ送信とメソッドの起動

メッセージ送信には、Scheme の procedure-call のシンタックスが使われ、procedure-call の形で実行される (RPC ではない)。

virtual クラス

virtual クラスのインスタンスは仮の値として扱われるオブジェクトであり、future や delay を評価した時に返される値として用いられるほかに、プロセス間の非同期通信にも有効である。

並列実行

PaiObject による並列実行は、spawn 式 (これは (make process '*E*) に相当) や、その他の PaiLisp 並列構文によるプロセスの生成によって行われる。プロセスを

PaiObject: A Parallel Object Oriented Language for PaiLisp
Yasuki IIZUKA, Takayasu ITO
TOHOKU University

制御するためには、suspend(プロセスの実行の中断)、process-kill(実行の強制終了) などのメソッドが用意されている。また、continuation によってプロセスの制御/同期が可能になっている。

3. PaiLisp による PaiObject の記述

言語の仕様を明確に与えるには、形式的な記述が望ましい。

PaiObject の言語仕様は、PaiLisp によって記述が与えられている。これは同時に、PaiLisp が並列オブジェクト指向言語を記述するのに十分な能力を持っていることをも示している。

PaiObject のオブジェクトの構造を、PaiLisp を用いて記述すると図1のようになる。オブジェクトは、lambda 式によるクロージャ(7-27) として表現できるが、これには、インスタンス変数環境(1-3)、クラスへのポインタ(5)、メソッド起動の機構(7-27)、排他機構(4-5,12-19) が含まれる。

この他に、PaiObject 特有のメソッド、virtual,process オブジェクトなどについても記述が与えられている。

```
 1: (let ((instv1  val)      ;; インスタンス変数環境
 2:       ....
 3:       (instvn  val)
 4:       (lock              ;; 排他機構
 5:        (exlambda (m args) (apply m args)))
 6:       (my-class <class>))    ;; クラスへのポインタ
 7:   (lambda (selector . args)  ;; オブジェクトの本体
 8:     (let ((method        ;; メソッド探索
 9:            (cond ((my-class search-method selector))
10:                  (else (error "No such method.")))))
11:       (if (exmethod? method)
12:           (lock            ;; 排他メソッドの適用
13:            (eval (method 'call)     ;; 環境操作
14:                  (append-environment ; 環境の結合
15:                   (method 'env)      ; メソッドの環境
16:                   (subset-environment
17:                    (current-environment)
18:                    (method 'ivars))))
19:            args)
20:           (apply           ;; 通常のメソッドの適用
21:            (eval (method 'call)
22:                  (append-environment
23:                   (method 'env)
24:                   (subset-environment
25:                    (current-environment)
26:                    (method 'ivars))))
27:            args)))))
```

このようにオブジェクトを表現した場合、メッセージ送信には次のような関数を用意しなければならない。

```
26: (define (send selector receiver . args)
27:   (apply receiver selector reciver args))
```

また、メソッドは次のような形をしているものとする。

```
28: (lambda (cmd)
29:   (cond ((eq? cmd 'call) ;; 実行される本体
30:          '(lambda <args> <body>))
31:         ((eq? cmd 'ivars);; 参照されるインスタンス変数
32:          <ivars>)
33:         ((eq? cmd 'env)  ;; 定義された環境を取り出す
34:          (current-environment))
35:         ((eq? cmd 'exmethod?)
36:          #f)))          ;; exmethod のフラグ
```

図1. PaiLisp による、PaiObject のオブジェクトの表現

4. 試作処理系の概要

設計した PaiObject について、KCL 上にその処理系を試作し、言語の記述能力の確認等を行っている。処理系は、複数の Lisp プロセスを一つの OS のプロセス上で動かす擬似並列処理系である。Lisp プロセスは、処理系によりタイムシェアリング的に実行される。これは、ある処理単位ごとに処理系がコンテキストスイッチングを行うことによって実現されている。スケジューリングはラウンドロビン方式を取っているが、タイムスライス(正確には処理単位の割り当て) を乱数でゆらすことにより、並列実行の非決定性をシミュレートしている。

現在、システムには言語インタプリタの他に、OS インターフェース、デバッガ、ライブラリ、プロセスモニタなどが含まれている。

図2に、PaiObject による5人の哲学者問題のプログラミング例の一部を示す。

5. 他言語との比較

並列オブジェクト指向言語として代表的なものに、ABCL/1[5],ConcurrentSmalltalk[6] などがある。

これらの言語は、「オブジェクト = プロセス」(並行オブジェクト) という形を取っており、オブジェクトが並列の単位になっている。オブジェクトは一時に一つのことしかできない。そして、メッセージの送信により暗黙のうちに並列実行が始まる。これらの言語は、分散環境用に設計されている。

一方、PaiObject は、共有メモリ型並列 Lisp である PaiLisp のために設計されていることから、このような形はとっていない。オブジェクトとプロセスは別で、明示的にプロセスを生成し、共有メモリを介した通信、1つのオブジェクトへの同時アクセスの許可などに特徴がある。

6. おわりに

前述の PaiObject の記述とは反対に、PaiObject による PaiLisp の記述も行われている。これは、メソッド定義式と最小限のメソッドからなる PaiObject セットを定義し、それを用いて PaiLisp-Kernel を含むすべての PaiLisp 並列構文を記述している。すなわち、PaiObject もまた、並列言語として十分な能力を持っていることを示している。

PaiObject は、共有オブジェクト、排他メソッド、プロセスクラス、プロセス操作メソッド等を明示的に取り入れたという意味で、並列 Lisp に対するオブジェクト指向言語の初めての試みといえよう。

```
 1: ;; 哲学者の行動
 2: (define start (make operation))
 3: (add-method (start (phil) self)
 4:     (loop (think self)
 5:           (get-fork self)
 6:           (eat self)
 7:           (release-fork self)))
 8: ;; フォークを取る部分
 9: (define get-fork (make operation))
10: (add-method (get-fork (phil num left right) self)
11:     (cond ((get-fork left self)
12:            (cond ((get-fork right self) #t)
13:                  (else (release-fork left self)
14:                        (get-fork self))))
15:           (else (get-fork self))))
16: (add-exmethod (get-fork (fork stat obj) self o)
17:     (cond (stat (set! stat '()) (set! obj o) #t)
18:           (else '())))
19: ;; メインプログラム
20: (define (5-phil)
21:     (let ((fork1 (make fork))
22:           :
23:           (fork5 (make fork)))
24:       (par (start (make phil fork1 fork2 1))
25:            :
26:            (start (make phil fork5 fork1 5)))))
```

図2. プログラミング例

参考文献

[1] Takayasu ITO, Manabu MATSUI : *A Parallel Lisp Language PaiLisp and its Kernel Specification*, SpringerLNCS No.441, 1990.

[2] Ress, J., and W. Clinger, eds., : *Revised³ Report on the Algorithmic Language Scheme*, ACM SIGPLAN Notices 21:12, Dec. 1986,

[3] Robert H. Halstead, Jr. *New Ideas in Parallel Lisp: Language Design, Implementation, and Programming Tools*, SpringerLNCS No.441, 1990.

[4] Kevin J. Lang, Barak A. Pearlmutter : *Oaklisp: An Object-Oriented Dialect of Scheme*, Lisp and Symbolic Computation, 1, 1988,

[5] 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

[6] 横手, 所 : 並行オブジェクト指向言語 *ConcurrentSmalltalk*, コンピュータ・ソフトウェア, Vol.2, No.4, 1988

† 本研究は科研費一般(A)01420029 の一環として行われたものである。