並行オブジェクト指向言語 y の通信機能

6J－3

崔梗洙* 小島一人** 野呂昌満** 原田賢一*
*慶應義塾大学理工学部 **南山大学情報管理学科

1 はじめに

分散オブジェクト指向計算モデルでは，オブジェクトを並行実行の単位として，オブジェクト間で一対一の通信を行ない，メッセージを受理することで対応する処理を行う．本稿では，ネットワークソフトウェアを記述することを主な目的として，従来のモデルで不十分であった点を拡張した計算モデル ψ と，これに基づくプログラミング言語 y について述べる．計算モデル ψ では，オブジェクト間通信を，一対一と一対多，通常と優先，同期と非同期，単方向と双方向の四種類に分類し，これらを任意に組み合わせて豊富な通信機能を表現可能とした．プログラミング言語 y ではこれらの機能を柔軟に組み合わせることで，ネットワーク環境で動作するプログラムの記述が容易に行える．

2 計算モデル ψ

分散オブジェクトを考慮した計算モデルとしては ABCM[Yon86] が有名である．ABCM ではオブジェクトはメッセージを受理することによってのみ活性化され，要求された処理を行った後休眠状態になる受動的なサーバ型オブジェクトである．

ネットワーク上に分散した資源を活用するようなプログラムを考えるとき，受動的なサーバ型オブジェクトだけではなく，他のオブジェクトに対して要求を出すだけの能動的なクライアント型オブジェクトが必要になる．

ψ のオブジェクトは並行実行のための計算能力と，

- 局所的記憶
- メソッド
- シナリオ

を持つ．シナリオは，メッセージの受理や送信を順路式の考えを取り入れて，オブジェクトの生成から消滅までの振舞いを記述したものであり，ABCM に比べても柔軟な表現ができる．シナリオを持つことで，能動的に動作するクライアント型オブジェクトや，サーバ／クライアントの両方の性質を持つハイブリッド型オブジェクトを考えることができる．

シナリオはオブジェクトごとに一つだけ存在するので，オブジェクト内での並列性は ψ では扱うことができない．

2.1 オブジェクト間通信

ψ では，オブジェクト間通信を次の四種類に分類して考えている．

- 単方向・双方向
- 一対一・一対多
- 同期・非同期
- 通常・優先

これらを組み合わせることで，さまざまな通信方法を考えることができるが，ここでは主に双方向の場合の同期・非同期の組合せについて考える．

単方向通信の場合は，非同期ならメッセージの送信後，送信側は相手の受信を待たずに実行を続け，同期ならメッセージの送信後，相手の受信を待って，実行を継続する．同期，非同期の間に双方向の場合ほどの違いはなく，単方向通信に同期を組み合わせることはメッセージの受信の確認程度の意味しかない．

双方向の交信の場合については，図 1のような組合せが考えられる．

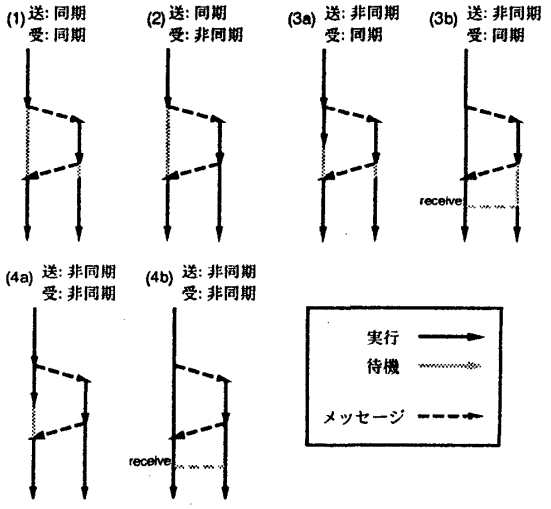

図 1: 双方向通信

Communicating feature of object oriented cuncurrent programming language y

Kyong-Rok Choi*, Kazuhito Kojima**, Masami Noro**, Ken'ichi Harada*
*Dept. of Computer Science, Keio University, 3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223, JAPAN
**Dept. of Information Systems and Quantitative Science, Nanzan University, 18 Yamazato-cho Showa-ku Nagoya 466, JAPAN

3 プログラミング言語 y

プログラミング言語 y[Koj93] は, 計算モデル ψ に基づいて, 主にネットワークソフトウェアの記述を目的とする並行オブジェクト指向言語であり, 構文は基本部分を C++ を用いる.

y のオブジェクトには, ψ のオブジェクトであり, 同時に実行される並行オブジェクトと, 単にデータ抽象を実現するための逐次オブジェクトがある. 逐次オブジェクトは, 単にデータ抽象を実現するためのオブジェクトであり, 並行実行されない.

並行オブジェクトは, その生成から消滅までの振舞いを記述したシナリオを持つ並行実行の単位で, ネットワーク上の異なるホスト上に分散することでネットワークソフトウェアを構成する. シナリオには, 他の並行オブジェクトとの同期, メッセージを受理するための条件の他に通常の文が記述できる. 図 2にシナリオの記述例を示す.

```
scenario{
  select{
    when(quelen < MAXQ){
      accept put();
    }
    when(quelen > 0){
      accept get();
    }
  }
}
```

図 2: シナリオの記述例

ただし, y の並行オブジェクトは単一のスレッドを持つだけなので, 自分自身へのメッセージを同期送信すると, 送信したメソッドがブロックしたままシナリオの実行が行われないので, デッドロックを招いてしまう.

y ではメッセージの送受信には, 以下のような構文が用意されている.

同期メッセージ通信 同期メッセージ通信の場合は, C++ と同様の関数呼出しの構文を用いる. 結果を必要としない単方向通信は, 送信者側で void 型に型変換することで示す.

```
<obj>.<message>({args})
```

send 文 send 文は非同期交信時に結果を格納すべきオブジェクトを指定して, メッセージを送信する.

```
send <obj>.<message>({args}) #<dest>;
```

receive 文 send 文で指定した結果の待ち行列から最初の結果を取り出す.

```
receive <dest>;
```

discard 文 send 文で指定した宛先に向けて送られて来た結果を破棄する.

```
discard <dest>;
```

accept 文 メッセージを受理する. return 文の終了により, メソッドの実行は終わり, シナリオの残りの部分の実行を継続する.

```
accept <message>({formal_arg_list});
```

return 文 結果を送信者に返す.

```
return {obj};
```

結果を必要とする非同期通信の場合, send 文の結果は宛先として指定されたオブジェクトに直接格納されるのではなく, そのオブジェクトに結びつけられた待ち行列が生成され, そこに到着順に格納される. receive 文は, この待ち行列から一つ値を取り出して指定されたオブジェクトに格納される. こうすることで, 一対多通信の場合に, 必要なだけの数の結果を送信側は利用したのちに, discard 文によって, 不要な結果を無視するとともに, 結果を格納する待ち行列を破棄することができる. discard された待ち行列に値が格納されようとすると, 結果を送信しようとしたオブジェクトに例外が送られる. 例外を受け取ったオブジェクトはそのメッセージに対する処理を中断することで, 次の処理を行うことができ, 不必要となった処理を続行する無駄が避けられる.

```
send obj_group.messsage(args) # dest;
    :
while(condition){
    receive dest;
    // do something on dest
}
discard dest;
```

図 3: 一対多双方向通信の結果の利用

4 まとめ

本稿では, 計算モデル ψ に基づくプログラミング言語 y の特に同期・非同期通信について述べた. ABCL/1 のスクリプトに対して, y の順路式の考えを取り入れたシナリオは, より柔軟な記述が可能である. また, 結果を格納するために単なる未来オブジェクトでなく, 待ち行列を導入したことで, 一対多通信における記述力が増した.

現在, C++ を用いて実装中であり, シナリオに相当する部分の仕組みはできているが, 待ち行列を用いて複数の結果を利用する部分は未実装の状態である.

参考文献

[Yon86] 米澤明憲, 柴山悦哉, Briot J. P., 本田康晃, 高田敏弘: 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 3, 3, 1986, pp. 9-23.

[Koj93] Kojima, K. and M. Noro: The Design of the Object-Oriented Concurrent Programming Language y, in *Proceedings of the Joint Conference on Soft. Eng. '93*, IPSJ, pp. 59-66.