オブジェクトシステムの形式的仕様記述に向けて *

6V－3

窪田歩　松本吉弘 †
京都大学工学部 ‡

1 はじめに

オブジェクト指向の研究において，近年オブジェクトの形式的仕様記述に関する研究が盛んに行われている [1, 2, 3]．オブジェクト指向のソフトウェア開発の，ラウンドトリップ型の開発形態においては，分析・設計・作成といったフェーズ間に頻繁にフィードバックがかかる．そのため，形式的仕様記述がインタプリタ上で記号的に実行可能であれば，より簡便に設計の不具合を見つけ上流工程にフィードバックをかけることができ，その有用性は高い．この目的に沿う言語としてFOOPS [1] とそのインタプリタ [4] 及びMaude [5] 等があるが，オブジェクト間の同期・非同期メッセージの扱いが不十分であり，並行性を持ったオブジェクトシステムの仕様記述法としては完全でない．

本研究は，代数的仕様とその項書換え系によるインタプリタをベースにした実行可能なオブジェクトシステムの形式的な仕様記述を目的とし，オブジェクトの内部状態 (属性値)，状態の更新，オブジェクト間の同期・非同期を含むメッセージ等の仕様記述法を提案し，インタプリタの実現にむけて，仕様によって表されるオブジェクトシステムの実行モデルを示す．

2 オブジェクトの内部状態とその更新

オブジェクト内で扱う属性値などのデータの仕様は代数的仕様によって与える．オブジェクトのクラスはオブジェクト識別子のソートとして扱う．オブジェクトの内部状態を表す属性値は，クラスをアリティーとする関数によって表される．この関数は，オブジェクトの状態によって様々に変わり得る．図1の例では，口座aの残高が300の時，balanceはbalance(a)= 300となる関数であるが，残高の増減によりbalance(a)の値は様々に変化する．オブジェクトの内部状態の変化はこの属性値関数の変更によって扱う．

オブジェクトはメッセージを受け取ることによって内部状態を変化させる．メッセージによる状態変化の仕様は，状態更新後の属性値に関する等式によって与える．この等式はそのまま属性値を更新する書換え規則と見なすことができ，項書換え系により実行することが可能である．

*Towards Formal Specification of Object Systems
†KUBOTA Ayumu, MATSUMOTO Yoshihiro
‡Faculty of Engineering, Kyoto University

```
class Account
  import
    INT .
  attributes
    balance : Account -> Int .
  messages
    deposit : Account Int -> Message.
    withdraw : Account Int -> Message.
  axioms
    balance(A)=M ->
      [deposit(A,N)] balance(A) = M+N .
    balance(A) = M and M > N ->
      [withdraw(A,N)] balance(A) = M-N .
  sending
    deposit(A,N) -> φ .
    withdraw(A,N) -> φ .
end class Account .
```

図 1: 銀行口座オブジェクト

3 オブジェクト間のメッセージ

オブジェクトはメッセージを受け取ることにより，その内部状態を更新するだけでなく，場合によっては1つまたは複数のメッセージを他のオブジェクトに対して送る．オブジェクト間のメッセージの送信形態には，過去型，現在型，未来型 [6] がある．これらはそれぞれ，過去型が非同期のメッセージ送信 (センダはレシーバからの返信を待たない)，現在型が同期メッセージ送信 (センダはレシーバからの返信を待つ)，未来型が同期と非同期の組み合わせである．並行性も考慮してオブジェクト間のメッセージを考えた場合，少なくとも過去型と現在型のメッセージ送信は扱えなければならないと考えられる．

3.1 従来の手法における問題点

メッセージの送信形態に関し，FOOPSでは同期型のメッセージしか扱っておらず，並行性の扱いには問題がある．FOOPSにおいてオブジェクトの状態の更新を行うメッセージ (FOOPSではmethod) は，状態が更新されたオブジェクトの識別子を返す関数であり，同期型のメッセージとなっている．

これに対しMaudeでは，"Concurrent Rewriting" を用いて並行オブジェクトシステムを扱っており，オブジェクト間では非同期のメッセージ送信が行われる．しかし，センダがレシーバからの返信を待つという意味での同期メッセージは扱っていない．

3.2 非同期メッセージの仕様記述

非同期メッセージは，アリティーに送信先オブジェクトのクラスを含み，ターゲットのソートがMessageであ

る関数として定義する．Messageは非同期メッセージを表すソートで，演算子 &(Message Message -> Message) と & に関する単位元 ϕ を持つ．メッセージによる状態の更新は2節で示したように，属性値に関するaxiomで定義する．

オブジェクトは非同期メッセージを受け取ると状態の更新を行なった後，場合によっては新しく，ひとつまたは複数の非同期メッセージを送信する．非同期メッセージの送信は以下のように記述する．

```
class Bank
  . . .
  messages
    transfer : Bank Account Account
      -> Message .
  sending
    transfer(B,A1,A2,M) ->
      withdraw(A1,M) & deposit(A2,M) .
end class Bank .
```

この例では口座間の転送を行うメッセージを受けた銀行オブジェクトが非同期メッセージによって2つの口座に引き出しと預金を依頼する．銀行オブジェクトは返り値を待たず，ただちに次のメッセージの受けつけが可能な状態になる．新しく非同期メッセージを送信しない場合は message(…) -> ϕ のように記述する．

3.3 同期メッセージの仕様記述

同期メッセージは，アリティーに少なくとも一つクラスを含み，ターゲットがMessage以外の関数とし，非同期メッセージと同じく状態の更新は属性値に関するaxiomで定義する．また返り値に関しても同様にaxiomで定義する．

```
  messages
    deposit_and_inquire : Account Int -> Int .
  axioms
    balance(A) = M ->
      [deposit_and_inquire(A,N)]
      balance(A) = M + N .
    deposit_and_inquire(A,N) = balance(A) .
```

上記の例において，2つ目のaxiomの右辺に現れるbalance(A)は，状態更新後の属性値を指す．属性値の観測関数も同期メッセージの一種と見なす．

同期メッセージの送信は，以下のようにaxioms部の等式の右辺に現れる．

```
  axioms
    · · · -> [· · ·]
      attribute(…) = SyncMessage(…) .
```

4 実行モデル

オブジェクトは同期メッセージを受けとった場合，必要とあれば他のオブジェクトに対して同期メッセージを送るなどして状態の更新を行なった後，値を返して実行を終了する．

非同期メッセージを受けとったオブジェクトは，同様に，必要とあれば他のオブジェクトに同期メッセージを送るなどして状態の更新を行ない，0個以上の非同期メッセージを送信して実行を終了する．sending部の記述を書換え規則として用いると，ある一つの非同期メッセージを起点とし，次々と非同期メッセージが発生・消滅し，最終的に ϕ となる非同期メッセージの系列を生成する．銀行の例では transfer(…)→ deposit(…)&withdraw(…) → ϕ という系列が生成される．

非同期メッセージの系列はそのまま並行オブジェクトシステムの大まかな制御の流れを表していると見ることが出来る．さらに詳しくオブジェクトシステムの挙動を見れば，一つの非同期メッセージが発生・消滅する間に，複数の同期メッセージがオブジェクトの状態の更新を行うために処理されている．

仕様によって記述されたデータの演算・状態の更新・メッセージの処理などは全て項書換え系によって実行可能であり，曖昧さのないオブジェクトシステムの形式的な仕様記述としてだけでなく，プロトタイピングにも用いることができる．

5 おわりに

本研究では，代数的仕様をベースにしたオブジェクトシステムの仕様記述法を提案し，状態の更新・メッセージの処理などに関する仕様の記述が，それらを書換え規則と見なすことにより実行可能であることを述べた．

今後の課題としては，インタプリタの実装が第一に挙げられる．またオブジェクト間の情報隠蔽・メッセージのブロードキャスト，自律プロセス等，本稿で未検討の課題もあり，今後の取り組みが必要である．

参考文献

[1] Gouguen, J. A. and Meseguer, J.: Unifying functional, object-oriented and relational programming with logical semantics, in *Research Direction in Object-Oriented Programming*, pp. 417–477, MIT Press (1987).

[2] Jungclaus, R. and Saake, G.: Formal Specification of Object Systems, in *TAPSOFT'91*, pp. 60–82, Berlin (1991), Springer, LNCS 494.

[3] Ehrich, H. D. and Gogolla, M.: Objects and Their Specification, in *8th Workshop on Abstract Data Types*, pp. 40–65, Berlin (1993), Springer, LNCS 655.

[4] Rapanotti, L. and Socorro, A.: Introducing FOOPS, Technical Report TR-28-92, Computing Laboratory, Oxford University (1992).

[5] Meseguer, J.: A Logical Theory of Concurrent Objects, in *ECOOP/OOPSLA '90*, pp. 101–115 (1990).

[6] 米澤明憲, 柴山悦哉：モデルと表現, 岩波講座ソフトウェア科学17, 岩波書店 (1992).