5N－3

代数仕様言語OBJによる並行分散システムの形式仕様作成法*

飯田 周作† 二木 厚吉‡
北陸先端科学技術大学院大学 情報科学研究科§

1 はじめに

並行分散システムは逐次に処理が行われるシステムと比較して、その性質を把握するのがはるかに困難である。これは並行分散システムでは処理の流れが一方向でないことによる。よって仕様段階で並行分散システムの性質を解析する事が出来るような形式仕様が望まれる。本研究の目的は並行分散システムに対する形式仕様を代数仕様言語を用いて作成する方法を提案することである。このために、Meseguerが示した書き換え論理を用いた並行オブジェクトモデルの形式化を、さまざまな性質を持つ通信路に対応できるように変更した。

2 代数仕様言語

本研究では代数仕様言語OBJ3[1]を採用した。OBJ3は順序ソート代数と等式論理に基づいて意味づけされており、項書換えシステムとして仕様を実行する事ができる。代数仕様言語において並行オブジェクトモデルを採り入れた例にはMeseguerによるMaude[2]がある。Maudeは並行項書換えを理論的ベースとしているが、本研究では並行でない項書換えをベースとしているOBJ3を用いてMeseguerの方法を応用した。

3 代数仕様での並行オブジェクトモデルの実現

Meseguerはオブジェクトとメッセージを要素とするBag(マルチセット)により構成されるConfigurationというものを考え、そのConfigurationを遷移規則により遷移させる事で並行オブジェクト計算を考えた。本研究ではConfigurationをオブジェクトの集合とメッセージのBagの組として定義し、メッセージのBagが並行分散システムにおける通信路であると考え、このConfigurationの定義を変更拡張する事によりさまざまな通信路を実現することを考える。実装時にはこのConfigurationにオブジェクトIDサーバというオブジェクトIDを管理するための機構を加えて3つ組とした。

4 仕様の記述方法

本研究で提案する方法では並行分散システムの仕様を以下の3つの部分に分けて記述する。

- 通信路の仕様 (Configurationの仕様)
- クラスの仕様
- 初期状態を構成するための仕様

通信路の仕様はConfigurationの定義である。これは次節"FIFOを満たす通信路の実現"で実際の記述例を載せる。クラスの仕様では、オブジェクトが持っている属性やその属性値、特定のメッセージに対する反応などが書かれる。初期状態を構成するための仕様はあまり重要でないので説明を省略する。以下に簡単な例を示す。

4.1 例

2つの実体から構成される分散システムを考える。この2つの実体は同一のクラスから生成され、属性としてGateとStatus、Lock持っている。Gateは属性値としてOpenとCloseを持ち初期値はClose、Statusは属性値として自然数を持ち初期値は0、Lockは属性値としてOnとOffを持ち初期値はOffである。この2つの実体間ではChangeとOpenGateという2つのメッセージがかわされ、OpenGateを受け取って属性LockがOffであれば属性GateをOpenにし、Changeを受け取って属性GateがOpenであれば属性Statusをインクリメントする。Statusがインクリメントが成功したら、もう一方のオブジェクトに対してOpenGateとChangeを送信する。属性GateがCloseの時Changeを受け取ると属性LockはOnになり、以後いかなるメッセージも受け付けなくなる。このシステムは無限にメッ

*Formal Specifications of Concurrent and Distributed Systems with Algebraic Specification Language OBJ

†Shusaku Iida (s_iida@jaist.ac.jp)
‡Kokichi Futatsugi (kokichi@jaist.ac.jp)
§Department of Information Systems,
Japan Advanced Institute of Science and Technology
15 Asahidai, Tatsunokuchi, Ishikawa, 923-12, JAPAN

セージのやりとりを繰り返しStatusが増加していくものとする。

はじめは通信路に関して何の制限も付けないので通信路の仕様は通常のConfigurationの定義のままである。クラスの仕様は以下のようになる。ただし、一部記述を省略した。

```
object FOO-CLASS is

*** 属性および属性値
  op Gate   : -> AId .
  op Open   : -> Value .
  op Close  : -> Value .

  op Status : -> AId .
  subsort Nat < Value .

  op Lock   : -> AId .
  op On     : -> Value .
  op Off    : -> Value .

*** 遷移規則

  eq trans
    { [ ([ O | Gate ; Close , Lock ; Off , A ]) OBJS ]
      [ (< O' | O | Change >) MSGS ]
      [ OIDS ] } =
    { [ ([ O | Gate ; Close , Lock ; On , A ]) OBJS ]
      [ MSGS ]
      [ OIDS ] } .

  eq trans
    { [ ([ O | Gate ; Close , Lock ; Off , A ]) OBJS ]
      [ (< O' | O | GateOpen >) MSGS ]
      [ OIDS ] } =
    { [ ([ O | Gate ; Open , Lock ; Off , A ]) OBJS ]
      [ MSGS ]
      [ OIDS ] } .

  eq trans
    { [ ([ O | Gate ; Open , Status ; N , A ]) OBJS ]
      [ (< O' | O | Change >) MSGS ]
      [ OIDS ] } =
    { [ ([ O | Gate ; Close ,
             Status ; (N + 1) , A ]) OBJS ]
      [ put-msg (< O | O' | Change >)
          (put-msg (< O | O' | GateOpen >) MSGS) ]
      [ OIDS ] } .
endo
```

5 FIFOを満たす通信路の実現

Configurationの拡張例としてメッセージの順序が保存されるような(FIFOが満たされている)通信路の実現を考える。

前節の例で作成した仕様はこのままではデッドロックを起こす。実際にこの仕様をOBJ3で実行させると、遷移規則の順番や初期状態の構成の仕方によっては、書き換えが途中で行きづまりデッドロックが検出される。FIFOが満たされているような通信路を考えれば、本来の仕様を得る事ができる。そのためにはメッセージを扱うデータ構造をBagからQueueに変更するだけで実現できる。変更した後の通信路の仕様は以下のようになる。ただしputはQueueに要素を追加するためのオペレータである。

```
object CONFIGURATION is
  protecting OBJECT .
  protecting MSG .
  protecting OID-SERVER .
  protecting SET-OBJECT .
  protecting QUEUE-MSG .
  sort Config .

  op {[_][_][_]} : Objects Msgs OId-Serv -> Config .
  op put-msg__ : Msg Msgs -> Msgs .

  var M : Msg .
  var MSGS : Msgs .

  eq put-msg M MSGS = put M MSGS .
endo
```

6 おわりに

本研究では代数仕様言語OBJ3で並行オブジェクト指向を扱う枠組を作り、その枠組を使ってさまざまな通信路を持つ並行分散システムの仕様が書ける事を示した。

OBJで作成した仕様は実行することによって検証が可能である。しかし逐次に処理が行われるようなシステムの仕様と異なり、並行分散システムでは実行に非決定性があるため、ある実行が成功したからといって必ずしもその仕様が望んでいる性質を持っているとは言えない。例えばデッドロックの検出などでは、実行してデッドロックが検出されれば、その仕様は必ずデッドロックを引き起こすパスを持つが、検出できなかったからと言って安全であるとは言えない。しかし、このように実行させてトレースを解析することが出来るということは並行分散システムの性質を仕様段階で明らかにする可能性が高まったと言えよう。

参考文献

[1] J. Goguen, T. Winkler, J. Mesegure, K. Fututsugi, and J.-P. Jouannaud. Introducing OBJ. Technical report, SRI International, Computer Science Laboratory, 1993.

[2] J. Meseguer. A logical theory of concurrent objects. In *ECOOP-OOPSLA'90 Conference on Object-Oriented Programming*, pages 101–115. ACM, 1990.

[3] J. Meseguer. A logical theory of concurrent objects and its realization in the maude language. In G. Agha, P. Wegner, and A. Yonezawa, editors, *Research Directions in Concurrent Object-Oriented Programming*. The MIT Press, 1993.

[4] J. Meseguer, K. Futatsugi, and T. Winkler. Using rewriting logic to specify, program, integrate, and reuse open concurrent systems of cooperating agents. Technical report, SRI International, Computer Science Laboratory, September 1992.