ソフトウェア工学 102-25
（1995．1．27）

並列オブジェクトシステムのモデル化と設計法

宮本 衛市† 渡辺 慎哉† 何 克清‡

†北海道大学工学部情報工学科 ‡武漢大学ソフトウェア工学国家重点研究実験室

実現すべきシステムが並列オブジェクトシステムを意図したものであったとしても、当初から並列オブジェクトが確定しているとは限らない。むしろ、並列オブジェクトを設定すること自体が設計の過程に含まれることであろう。そこで、我々は、システム仕様が並列オブジェクトを想定した振舞いの型で与えられているとき、この型をさらに詳細化し、与えられた問題環境のもとで並列オブジェクトを設定し、基本的な振舞いの型に対応したイベントトレースを関係するオブジェクト間で作成し、各オブジェクトの振舞いの型を確定した上でオブジェクトのプログラミングを行うような設計方略について検討している。本稿では、簡単なスタックの例を通して、並列オブジェクトシステム設計に対する我々の考え方を述べている。

An Analysis of Behavior of Composite Objects Based on Type

Eiichi Miyamoto† Shin-ya Watanabe† Keqing He‡

†Division of Information Engineering, Faculty of Engineering, Hokkaido University
Kita 13 Nishi 8, Kita-Ku, Sapporo 060, Japan
‡State Key Laboratory of Computer Software Engineering, Wuhan University
Wuhan 430072, People's Republic of China

Although a system is to be implemented on the basis of concurrent objects, it is not always decided from the beginning what they are, rather included in the design process. Consequently, when the system specification is given by the type based on behaviors of concurrent objects which the system might be consist of, we are now considering design strategies to refine the given type, to set up concurrent objects under the given environment of the problem, to make out event traces between related objects for the basic types of behaviors, to decide the type of the behavior for each object, and to program it. In this paper, we report the idea for the design of concurrent object systems through a simple design example of a stack.

1 はじめに

実現すべきシステムが並列オブジェクトシステムを意図したものであったとしても、当初から並列オブジェクトが確定しているとは限らない。むしろ、並列オブジェクトを設定すること自体が設計の過程に含まれることであろう。我々は各並列オブジェクトが正規表現で表されるような状態遷移をするものとした上で、並列オブジェクトの振舞いを表す型を定義し、さらに並列オブジェクト群を統合した並列オブジェクトシステムとしての振舞いの型を定義した[1]。

そこで、我々はシステム仕様が並列オブジェクト群を想定した振舞いの型で与えられているとき、その仕様を基にした並列オブジェクトシステムの設計方略を検討している。その大まかな枠組は以下の通りである。まず、与えられたシステム仕様とシステム環境から、このシステム仕様を満足させる並列オブジェクトを選定し、各オブジェクトの仕様を振舞いの型で設定する。もし、一つのオブジェクトとして大き過ぎるオブジェクトがあれば、それをシステム仕様とみなして、上記の手続を繰り返す。システム仕様が並列オブジェクト群に展開されると、振舞いの各基本型に対して、関係するオブジェクト間でイベントトレースを作成し、これらに基づいてオブジェクトの内部設計を行う。

オブジェクト指向方法論としてのOMTでは、その動的モデルでイベントトレースを用いてシステムのシナリオを表すことを謳っているが、しかしこれは設計者の直観に訴えるだけで、具体的な設計上での指針とはなっていない[2]。我々はイベントトレースで表されるようなオブジェクトの振舞いを構造化し、これに基づいて並列オブジェクトの設計の枠組を模索しており、今回その試案を提案する。

以下、2節では並列オブジェクトに対する振舞いの型に関する概要を述べ、3節ではスタックを例に取って並列オブジェクトの設計を行う。4節では考察を含めた結論と今後への展望を述べる。

2 オブジェクトの振舞いに基づく型

オブジェクトに対する静的な型としては、オブジェクトが受け付けるメッセージ集合に基づく型などが考えられるが、我々はオブジェクトの動的な枠組として振舞いに基づく型を定義した。これは以下のような考え方に基づくものである。

(1) 並列オブジェクトの状態遷移は正規表現に基づくパス式で表すことができるものとする。

(2) 正規表現で表されている各オブジェクトのパス式を、逐次的な状態遷移のみからなる基本状態遷移の集合に分解する。

(3) 各オブジェクトの基本状態遷移間でオブジェクト群の基本協調動作を求め、これをオブジェクトの振舞いに対する基本型と定義する。

(4) 各オブジェクトの基本型の集合を、パス式の構造に対応させた構造型へ統合する。

(5) 各オブジェクトの構造型を統合して、並列オブジェクトシステムとしての振舞いに基づく型を求める。

いま、例として図1に示すような状態遷移をもつ4つのオブジェクトを考える。これらの状態遷移を分解して得られた基本状態遷移間に、図2の点線で示すような$F_1,...,F_5$の協調動作があるものとすると、各オブジェクトの振舞いによる型は次のように表すことできる。

$$O_1: \ (F_1+F_2)\cdot(F_3\cdot F_4^*)^* \qquad (1)$$

$$O_2: \ (F_1+F_2)\cdot(F_3\cdot F_4^*)^* \qquad (2)$$

$$O_3: \ (F_1+F_2)\cdot F_5^* \qquad (3)$$

$$O_4: \ (F_1+F_2)\cdot F_5^* \qquad (4)$$

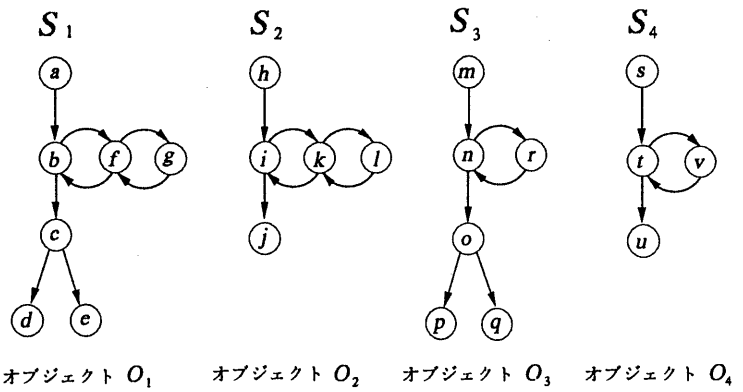

図 1: オブジェクトの状態遷移の例

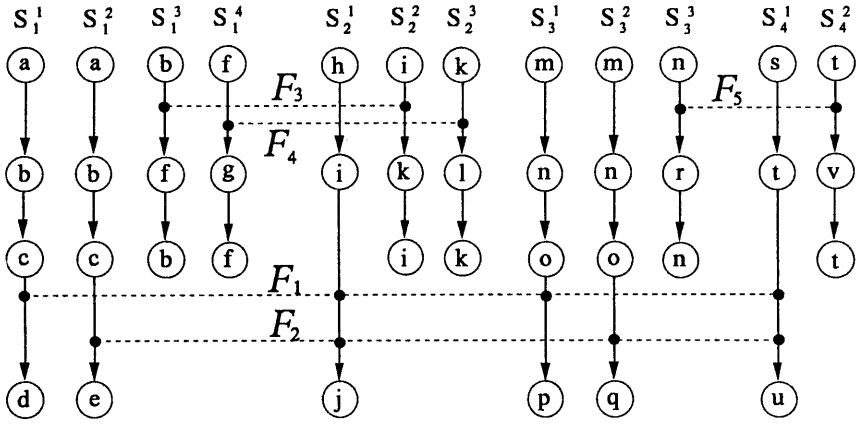

図 2: 基本状態遷移とそれらの間での協調動作

ここで、基本協調動作は振舞いの基本型に対応するので、$F_1, ..., F_5$は協調動作に対応した型も表すものとする。上式で、例えばオブジェクト O_1はF_1, F_2, F_3, F_4の協調動作に関与し、排他的協調動作F_1とF_2を主動作として、それらが2重の入れ子になった協調動作を抱えていることを示している。

各オブジェクトの振舞いを統合した並列オブジェクトシステムとしての振舞いによる型は次のようになる。

$$(F_1 + F_2) \cdot ((F_3 \cdot F_4^*)^* \parallel F_5^*) \qquad (5)$$

上式は排他的協調動作F_1とF_2を主動作として、入れ子になった2つの並行動作があり、一方はさらに内部動作を抱えていることを示している。

3 並列オブジェクトの設計

前節の説明では、振舞いによる型の概念を導出するために、まず並列オブジェクトの存在を前提としてそれらの振舞い、さらにはシステムとしての振舞いを考察し、その形式的記述を試みた。しかし、並列オブジェクトシステムを設計するときには、むしろこの逆順の手続きを経なければならない。すなわち、まずシステム仕様が与えられ、これから必要な並列オブジェクトが設定され、各オブジェクトに動作仕様が与えられる。必要なら、これらオブジェクトはシステムとみなして、さらに並列オブジェクト群へと詳細化を進めていく。そこで、ここではスタックを例に取り、上記の設計方略を説明する。

図3はスタックとして与えられた仕様であり、Push-in 動作Pと Pop-up 動作Qが並行動作するものとしてスタック仕様Sが規定されているものとする。

$$S = P^* \parallel Q^* \qquad (6)$$

もし、スタック容量を指定するときには、その容量をNとして、次のように与えることもできる

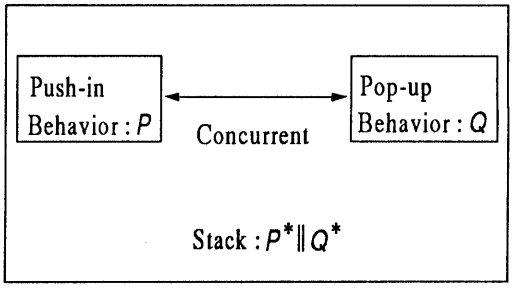

図 3: スタックの振舞いの型による仕様

[3]。

$$S' = P\{p\}^* \parallel \{0 \le p - q \le N\} \; Q\{q\}^* \qquad (7)$$

ここで、pおよびqはそれぞれ協調動作PおよびQの実行回数を表すものとする。上式はPとQの並行実行に制約を与えるものである。

PとQが並行動作することから、Pを構成するオブジェクト群とQを構成するオブジェクト群に共通するオブジェクトを置くことはできない。そこで、図4に示すように、Pはオブジェクト Producer と Push-in Manager 、Q はオブジェクト Consumer と Pop-up Manager から、それぞれ別個に構成されるものとする。なお、stack を操作するときの相互排除制御および (7) 式で与えられた制約の履行は別途行われるものする。

次に、PとQの詳細化を考える。Push-in 動作Pはスタックが満杯のときにはスタックに余裕ができるまで待たねばならず、一方、Pop-up 動作はスタックが空のときにはデータが入ってくるまで待たなければならない。それゆえ、PとQをそれぞれ次のように詳細化する。

$$P = P_1 \cdot P_2^* \qquad (8)$$

$$Q = Q_1 \cdot Q_2^* \qquad (9)$$

ここで、P_1とQ_1はそれぞれStack-in と Pop-up を試みる動作であり、P_2とQ_2はそれぞれStack-in と Pop-up ができるようになるまで待機を続ける動作を表すものとする。

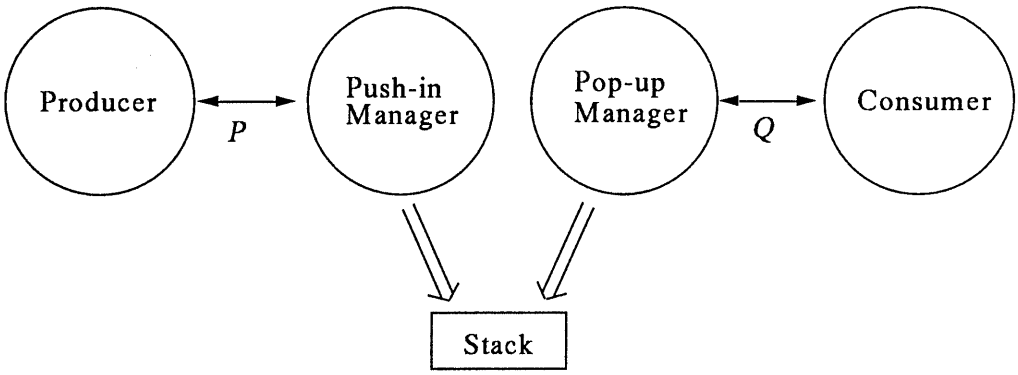

図 4: 並列オブジェクトによるスタックの実現

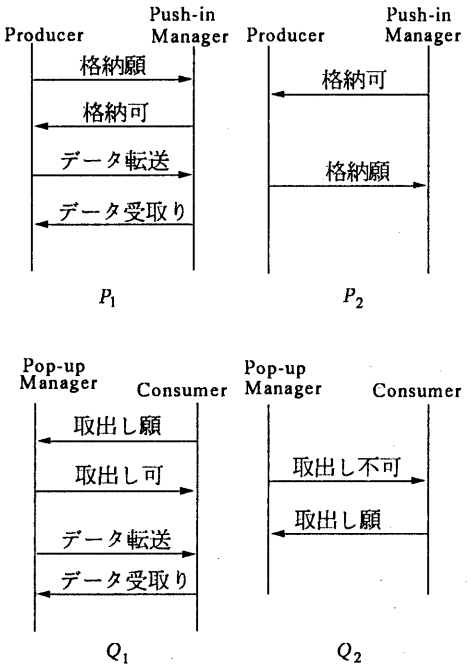

図 5: 協調動作のイベントダイアグラム

必要があれば、さらにオブジェクトまたは協調動作の詳細化を続けていくが、今回の例は簡単であるので、これで詳細化の必要はないと判断する。そこで、図5に示すような各協調動作のイベントトレース図を作成する。このトレース図が各オブジェクトの内部設計の、いわば仕様となる。オブジェクト間でのイベントはイベントトレース図に従って発生するのであって、その意味ではこのイベントトレース図はイベントの発生順序に対する制約を表していることになる。この図で、各オブジェクトの時間軸を表している縦線のメッセージの授受の間は、オブジェクトのある状態を意味しており、これに名前を与えればパス式を求めることができる。

一方、相互排除制御とスタックの容量制限に関しては、オブジェクト Push-in Manager 、Pop-up Mnager および Stack との間の協調動作 W として、次のように考える。すなわち、図6に示すように、Push-in Manager と Stack との協調動作を U、Pop-up Manager と Stack との協調動作を V とすると、W は

$$W^* = (U + V)^* \qquad (10)$$

と表すことができる。上式は U または V が不可分かつ排他的に動作することの繰り返しであること

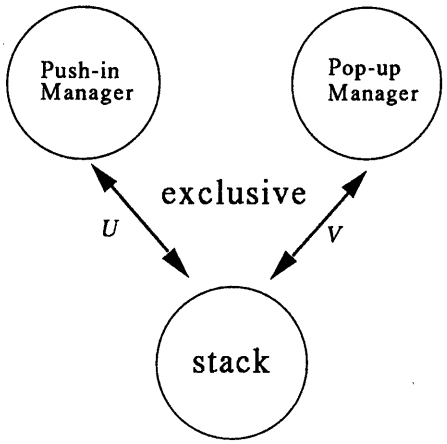

図 6: スタックの相互排除制御

を意味する。

さらに、Uにはスタックに余裕があった場合の協調動作U_1と、満杯の場合の協調動作U_2があり、一方Vにはスタックにデータがあった場合の協調動作V_1と、スタックが空の場合の協調動作V_2があり、したがってWは

$$W^* = (U_1 + U_2 + V_1 + V_2)^* \qquad (11)$$

と詳細化することができる。上式の各協調動作のイベントトレース図は図7のようになり、どの動作が排他的に実行されるかは非決定的である。

4 おわりに

並列オブジェクトシステムを振舞いの型に基づく仕様から設計していく方略を簡単なスタックを例に取って説明したが、まだ基本的なアイディアの段階であり、いろいろな問題点が残されている。第1には、何といっても本格的なシステムを設計してみて、ここでの考え方が大規模なシステム設計に耐えうるものなのかどうかを実証してみることである。その他には、排他的あるいは並行な協調動作があった場合、段階的な詳細化がいつも可能であるかどうか、排他的な動作は非決定的な記

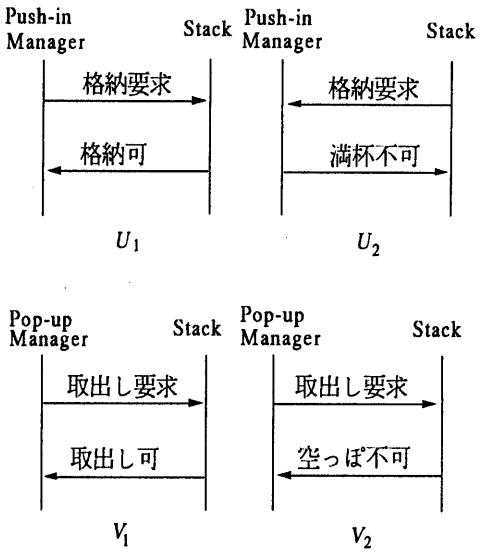

図 7: 相互排除でのイベントダイアグラム

述にならざるをえないが、オブジェクトの内部設計に問題はないかなど、検討する必要がある。

現在、オブジェクトの振舞いに基づく記述がそのままできるオブジェクト指向言語のプロトタイプを作成中であり、より高度なオブジェクト記述の可能性を探っている。

参考文献

[1] 何, 渡辺, 宮本: 並行オブジェクト群による協調動作に対する型の定義, 情報処理学会論文誌, Vol.36, No.2 (1995).

[2] ランボー,J. 他: オブジェクト方法論OMT, トッパン (1992).

[3] 上居, 渡辺, 宮本 : Predicate Path Expression に基づく並行オブジェクトの振舞いの型の定義, 日本ソフトウェア科学会第 11 回大会論文集, pp.137-140(1994).