Vol. 36 No. 2 情報処理学会論文誌 Feb. 1995

並行オブジェクト群による協調動作に対する型の定義

何 克清[†] 渡辺慎哉[††] 宮本衛市[††]

これまでソフトウェアシステムを分析・設計するために，オブジェクト指向モデルがもつ優れた能力を活用した強力な方法論が展開されてきた．オブジェクトのもつカプセル化能力により，ソフトウェアシステムを部品の組立で構成していくことができる反面，オブジェクト群としての協調動作の把握が難しい．それゆえ，オブジェクト群の振舞いの解析，あるいは振舞いの記述は，オブジェクト群の協調動作を陽に表現する上できわめて重要である．われわれは，オブジェクトの状態遷移が正規表現で表されるものとして，オブジェクトの振舞いを分析し，オブジェクト群の協調動作を構造的に定義する型を提案する．この型に基づきオブジェクト群の振舞いを記述すれば，オブジェクト群の振舞いの理解が容易になるばかりでなく，この型によりオブジェクト群の振舞いに関する静的および動的な整合性の判定ができるようになるものと考えている．

Definition of a Type for Cooperative Behaviors Based on Concurrent Objects

KEQING HE,[†] SHIN-YA WATANABE[††] and EIICHI MIYAMOTO[††]

We propose a type defining the cooperative behaviors of objects on the assumption that the state transitions of objects can be expressed by regular expressions. Behaviors of objects are decomposed into basic state transitions which are composed of only basic states. Cooperative behaviors among concurrent objects are considered to be constructed through basic state transitions of each object. We define a behavioral type as the event relation among the basic state transitions, and a structured type for structured cooperative behaviors. We consider that we can easily understand the behaviors of objects, and decide the cooperabilities of objects, through the description of the types of objects.

1. はじめに

高品質で，かつ頑健なソフトウェアを構築するために，ソフトウェア・プロトタイピングはきわめて有効な手法と考えられている．ソフトウェア・システムの要求仕様を実証するために，計算機を利用してソフトウェア機能をシミュレートすることにより，ソフトウェアの振舞いを直接確認することができ，ユーザニーズと要求仕様との間のくい違いを未然に防ぐことができる．特に，ソフトウェア・システムが並行プロセスを含むようなリアクティブ・システムのような場合には，各プロセスが互いに絡み合い，システムとしての挙動を大変に複雑にしており，そのためプロトタイピングによるシステムの動作確認は必須といえよう．

† 武漢大学ソフトウェア工学国家重点研究実験室
State Key Laboratory of Computer Software Engineering, Wuhan University
†† 北海道大学工学部情報工学科
Division of Information Engineering, Faculty of Engineering, Hokkaido University

ソフトウェア・システムをプロトタイプするためには，ソフトウェアをその挙動に着目して抽象的に表現するモデルが必要である．オブジェクト指向モデルはソフトウェア技術において重要な概念である情報隠蔽，抽象データ型，継承，委譲などを包含し，そのためソフトウェア部品の基本モデルとしても勝れており，ソフトウェア実現の基本モデルとして近年大いにもてはやされている．OMT はオブジェクト指向モデルに基づくソフトウェアのモデル化と設計の，いわば集大成ともいえる方法論である[1]．OMT はオブジェクトモデル，動的モデルおよび機能モデルから構成されており，システムをそれぞれの視点からモデル化することを意図している．このうち，オブジェクトモデルはシステムの静的で構造的な側面を捉え，機能モデルはシステムの変換的で関数的な側面をとらえ，動的モデルはシステムの時間的で動作的な側面を捉えようとしている．前2者はいずれもシステムの静的な分析が対象となるのに対し，動的モデルはシステムの時間的振舞いを対象としている．OMT の動的モデルで

は，オブジェクトの動的な振舞いを記述するモデルを与えているが，さらに突込んでシステム内で時間的にどのようなイベントが発生し，それが各並行オブジェクトをどのように制約し，オブジェクト間にどのような因果関係をもたらしているのかを構造的ないしは形式的に分析し，表現する手段を用意していない．

そこで，本論文では並行オブジェクトの振舞いに構造的な枠組を設定し，これをオブジェクトの振舞いに対する一種の型と見なし，オブジェクトの振舞いの仕様記述のモデルとすることを提案する．そのために，並行オブジェクトの振舞いを正規表現からなるパス式で表し，さらにこのパス式を逐次遷移からなる基本状態遷移の集合に分解し，各オブジェクトの基本状態遷移間で協調動作を定義する．その上で，この協調動作で引き起こされるイベント群の順序集合をオブジェクト群の協調動作に基づく型と定義する．これをオブジェクトの振舞いに対する原始型と考え，各オブジェクトのパス式に基づいて各オブジェクトの振舞いに関する構造型を合成していく．さらに，オブジェクト群全体を1つのシステムとして見たとき，そこで展開される協調動作の構造型も定義する．これにより，オブジェクトは機能的な仕様記述のほか，時間的な振舞いに対する仕様記述も形式的に行うことができ，オブジェクトの仕様記述が一層向上するものと思われる．

以下，2章では並行オブジェクトの振舞いに対する型を導入し，さらにシステム全体としての協調動作に対する型を定義する．3章では，2章で定義した型を共通問題である在庫管理システムの問題に適用し，各オブジェクトの振舞いの型およびオブジェクト群全体としてのシステムの協調動作の型の定義を行い，型により振舞いの仕様記述が簡潔かつ形式的に行えることを示す．4章では，オブジェクトの振舞いの解析に対する他の研究例との比較を行う．5章は結論である．

2. 並行オブジェクトの振舞いに対する型の導入

本章では，オブジェクトの内部状態の遷移式を基本とし，オブジェクトの振舞いに対する型を定義する．

2.1 オブジェクトの型

オブジェクトを特徴づけている概念としてクラスがある．一般に，同一のクラスから導出されたインスタンス・オブジェクトは同じデータ構造とメソッドを持つ．しかし，この性質はオブジェクトの有する機能的側面のみを形式化したものであって，オブジェクト本来の振舞いはオブジェクト間のメッセージのやりとりを通したオブジェクト操作の記述の中に折り込まれている．分散環境に適応させるため，メッセージの集合でオブジェクトの型を直接定義することも提唱されているが[2]，振舞いに関してはクラスと同様，陽に表現することは考えていない．オブジェクト間でどのようなメッセージの授受が行われ，それが時間的にどのように進行するかは，OMT の動的モデルでも用いているように，イベントトレースを用いて表すことができる．しかし，このイベントトレースはオブジェクトの振舞いの直観的理解のために用いられるものであり，オブジェクトの振舞いに何ら制約を与えるようなものではない．このような状況のもとでは，オブジェクト指向モデルは単にオブジェクト間ではメッセージのやりとりで計算が進行していくというだけで，計算の進行に関し構造化された枠組を何ら有していない．

そこで，オブジェクトの振舞いに制約を課すいろいろな枠組が提案されてきた．その主たる目的は並行オブジェクト間での協調動作に伴う各オブジェクトの振舞いの因果関係を明確にし，そこに制約を課すことを目指してきた．それゆえ，これらはオブジェクトの，いわば局所的な振舞いの掌握を図っているのに対し，われわれは並行オブジェクト群をシステムの構成要素として，システムの大局的な振舞いを構造的に記述することを考えている．

2.2 パス式に基づくオブジェクトの振舞いの解析

一般に，オブジェクトは内部変数で実現される内部状態を持つことができる．そこで，この内部状態を問題の分析レベルに応じて，いくつかの状態に分割して考えるものとする．オブジェクトは分割されたいずれかの状態にあり，受信したメッセージの種類とそのときの状態により決まる動作を行う．この動作は他のオブジェクトへのメッセージの送信となって現れ，自身は次の状態へと遷移する．

このように，オブジェクトを一種の有限状態機械とみなし，その振舞いをパス式で表現することを考える．パス式は並行プロセスの同期制御[3]や，プログラムのデバッグ[4]などで，プロセスやプログラムの振舞いを先験的に与えるために考えられた記法である．本論文では，このパス式を簡単のため正規表現で表すものとし，パス式の構文規則を次に示す．

$$path ::= \{loop | branch | state\}^{+} \quad (1)$$

$$loop ::= \text{'['} loop\_cond \text{','} loop\_body \text{']'} \quad (2)$$

$$branch ::= \text{'}\langle\text{'} branch\_cond \text{','} path1 \text{','} path2 \text{'}\rangle\text{'}$$

(3)

*loop_body, path*1, *path*2 : := *path*　(4)

loop_cond, branch-cond : := *state*　(5)

state : := *character_string*　(6)

ここで，$^+$は1回以上の繰り返しを，|は選択を表すメタ記号であり，' 'で囲まれた記号は終端記号である．*state* はオブジェクトの基本状態の名前を表し，[] で囲まれた2つ組はループを表すパス式であり，〈 〉で囲まれた3つ組は分岐を表すパス式である．

図1はオブジェクトの状態遷移の例を示したものである．パス式ではイベントに着目して式を表現するが，オブジェクト分析や設計においてはオブジェクトの状態に着目したほうが一般に考えやすいので，ここではオブジェクトの状態遷移に着目してオブジェクトの振舞いを表すことにする．図1のパス式は次のように表される．

$$T=a[b,[f,g]]\langle c,d,e\rangle \quad (7)$$

上式で，例えば $[f,g]$ は $fgfg\cdots f$ を表し，状態 f が繰り返しの継続あるいは終了のメッセージを受け取り，g がループ本体であることを意味する．また，$\langle c,d,e\rangle$ は状態 c が受け取るメッセージにより，パス式 d または e に遷移することを意味する．なお，デフォールトとしてパス式の右辺全体は繰り返されるものとする．

次に，一般には正規表現で表される状態遷移を基本状態の逐次的な状態遷移のみからなる基本状態遷移の集合で表すことを考える．そのため，パス式の構文規則に基づき，与えられたパス式および派生するパス式に次の分解規則を適用する．

(1) *PATH* をパス式の集合とし，最初は与えられたパス式のみを含むものとする．

(2) *PATH* に含まれるすべてのパス式に対し，(1)式で定義された *path* を構成する *loop* がいくつか存在するとき，*loop_cond* のみを残して *loop* を取り出し，*loop_cond loop_body loop_cond* からなるパス式を *PATH* に追加する．

(3) *PATH* に含まれるすべてのパス式に対し，(1)式で定義された *path* を構成する *branch* がいくつか存在するとき，*branch_cond* のみを残して *branch* を取り出し，各 *branch* の *path*1 と *path*2 のすべての組合せからなるパス式を *PATH* に追加する．

(4) *PATH* に含まれているパス式に *loop* や *branch* がすべて除去されるまで，(2)または(3)を再帰的に適用する．

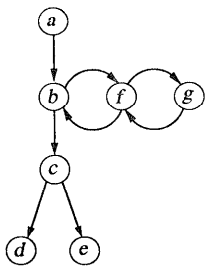


図 1　オブジェクトの状態遷移の例
Fig. 1　An example of the state transition of an object.

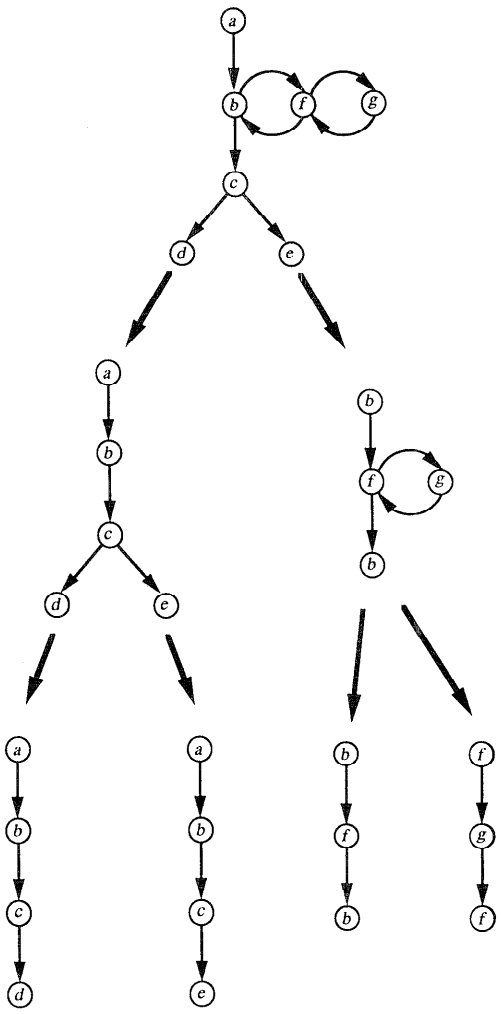


図 2　パス式の分解の例
Fig. 2　An example of the decomposition of a path expression.

この分解規則により最終的に *PATH* には，*loop* や *branch* を持たず，*state* の並びのみで構成されるパス式の集合が生成される．そこで，基本状態 *state* のみの逐次遷移からなるパス式を基本パス式と呼ぶことにする．**図2**は，図1に示すオブジェクトの状態遷移を分解していくプロセスを表したものである．その結果，(7)式のパス式は次の基本パス式で表す基本状態遷移の集合に分解することができる．

$$\{abcd, abce, bfb, fgf\} \qquad (8)$$

オブジェクトのすべての振舞いがパス式で表されているものとすると，オブジェクトはパス式から導出された基本状態遷移のいずれかを遷移中である．

2.3 基本状態遷移間での協調動作

各オブジェクトは互いにメッセージの授受を行いながら協調的に動作していくが，それぞれのメッセージは完全に独立ではなく，意味的にまとまった一連のメッセージ群を構成する．そこで，各オブジェクトがこのメッセージ群をやりとりしながら行われる一続きの動作を，ここでは協調動作と呼ぶ．

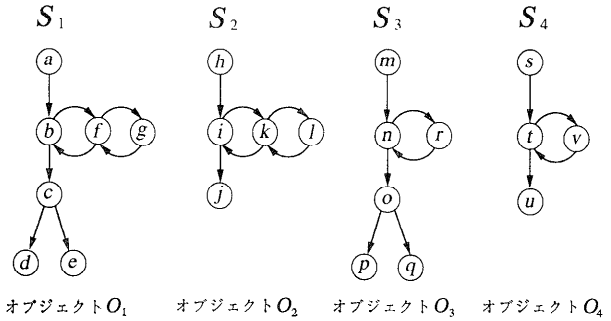


図3 4つのオブジェクト状態遷移
Fig. 3 State transitions of four objects.

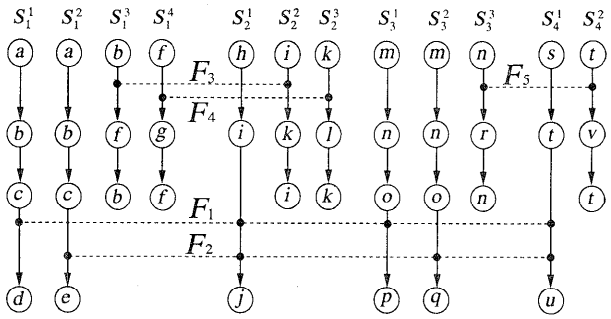


図4 オブジェクト間での協調動作
Fig. 4 Cooperative behaviors among objects.

各オブジェクトはそれぞれの基本状態遷移のいずれかの上を遷移しながら，対応する協調動作に関与していくが，オブジェクトの状態遷移と協調動作の間には，次のような性質がある．

[性質] あるオブジェクトが，ある基本状態遷移にあるとき，他のオブジェクトはたかだか1つの基本状態遷移上でしか，それとの協調動作に関与できない．(基本状態遷移の排他性)

[証明] もし，あるオブジェクトの1つの基本状態遷移に対して，他のオブジェクトが2つ以上の基本状態遷移で対応していると仮定すると，それは2つ以上の繰り返しの共存，あるいは分岐に伴う排他的な状態遷移の共存を意味し，前者のオブジェクトでは対応できず，システム全体の振舞いとしては矛盾する．

本論文で述べる協調動作とは，これに関与する複数の並行オブジェクトの基本状態遷移の集まりで構成される協調的な動作のことであり，これを並行オブジェクト群からなるシステムの基本的な機能単位として位置づける．上記の性質により，各協調動作は互いに独立であるとみなせるため，協調動作同士が干渉することがない．さらに，オブジェクトはいずれかの基本状態遷移上にいるので，それに対応した協調動作に関与していることになる．**図3**は4つのオブジェクトの状態遷移を表しており*，**図4**はそれぞれを基本状態遷移に分解し，オブジェクト間の協調動作を通して，上記の性質を概念的に説明したものである．ここで，$S_1, \cdots, S_4$ はオブジェクト $O_1, \cdots, O_4$ のパス式であり，各 S の上つきの添字で基本状態遷移を区別している．$F_1, \cdots, F_5$ はこれらオブジェクト間で行われる5つの協調動作を表しており，点線で示す基本状態遷移間でメッセージの授受が行われているものとする．なお，S_2^1 と S_4^1 は F_1 と F_2 に関与しているが，これらは異なる協調動作であるため，本節で述べた性質には矛盾しない．

2.4 協調動作に基づく型の定義

前節で述べたことを言い換えると，ある局面でいくつかのオブジェクトが，しかもそれぞれがある1つの

* これらオブジェクトに対し，S_1 と S_2 で2次元データの処理を，S_3 と S_4 で1次元データの処理をそれぞれ行い，これらの結果を持ち寄って S_1 と S_3 で最終処理を行うような問題を想定することができる．

基本状態遷移を行いながら，協調してオブジェクト群としての振舞いを構成しているということができる．いま，図4に示すような協調動作の1つに着目したとき，**図5**のイベントトレースで示すように，基本状態遷移を引き起こすメッセージ群をユニークに割り付けることができ，その協調動作にかかわるイベント群はそれらの生起順序から**図6**に示すような半順序関係を構成することになる．ここで，m_i $(i=1,\cdots,8)$ はメッセージの発信およびそれに伴う受信を意味する内部イベントを表し，m_I および m_O は，それぞれ協調動作を引き起こす外部イベントと，協調動作の結果として外部へ送られるイベントを表す．また，イベントをカッコで囲んでメッセージの放送に対応したイベントと，同期に対応したイベントを表す．この半順序関係はイベントの前後関係を要素とする次のような集合で表すことができる．

$$V=\{m_I\rightarrow(m_1,m_2),\ m_1\rightarrow m_3,\ m_2\rightarrow m_5,\ m_3\rightarrow m_4,\ m_5\rightarrow m_6,\ m_4\rightarrow m_7,\ m_6\rightarrow m_8,\ (m_7,m_8)\rightarrow m_O\} \quad (9)$$

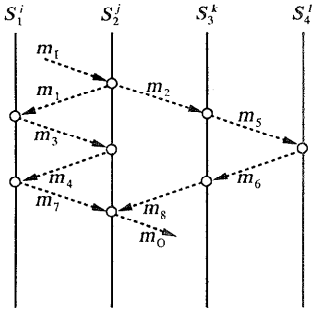


図5　協調動作のイベントトレースの例
Fig. 5　An example of the event trace of cooperative behaviors.

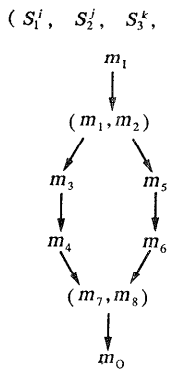


図6　イベント群の半順序関係
Fig. 6　A partially ordered relation of events.

一方，各オブジェクトからみると，そのオブジェクトが関与するイベントのみがそのオブジェクトの振舞いを決めることになる．基本状態遷移では状態の遷移が確定しており，各オブジェクトに関するイベント群は全順序関係になる．**図7**は各オブジェクトの振舞いを示したものであり，オブジェクト $O_1,\cdots,O_4$ に対応して，それぞれ次のような全順序集合となる．

$$V_1=\{m_1\rightarrow m_3,\ m_3\rightarrow m_4,\ m_4\rightarrow m_7\} \quad (10)$$

$$V_2=\{m_I\rightarrow(m_1,m_2),\ m_1\rightarrow m_3,\ m_3\rightarrow m_4,\ m_4\rightarrow m_7,\ (m_7,m_8)\rightarrow m_O\} \quad (11)$$

$$V_3=\{m_2\rightarrow m_5,\ m_5\rightarrow m_6,\ m_6\rightarrow m_8\} \quad (12)$$

$$V_4=\{m_5\rightarrow m_6\} \quad (13)$$

ここで，図6および図7の意味を考えてみる．図7は各オブジェクトの基本状態遷移に対応したイベント列であり，図6は協調動作を引き起こすイベント群の関係を表している．この協調動作を引き起こすイベントの半順序関係は，一般にラティス構造になり，並行オブジェクト群がもつ基本的な機能に対応した枠組を表している．それゆえ，並行オブジェクト群の協調動作に対し，次のような型を定義する．

[定義1]　ある協調動作を構成するイベント群の半順序関係 V をその協調動作の型*と定義する．

この型は基本状態遷移に基づく不可分な協調動作に対するイベント群の構造を与えるものであり，それゆえ協調動作に対する原始型を与えるものである．

一方，図7に示すような各オブジェクトにおけるイベントの全順序関係 $V_1,\cdots,V_4$ は，協調動作に関与する各オブジェクトの枠組を表しており，これらは半順

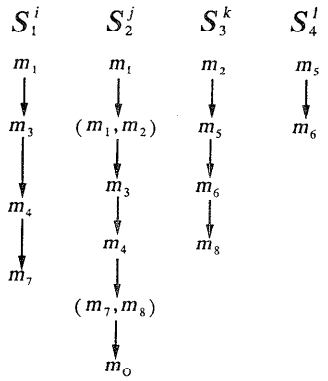


図7　各オブジェクトにおけるイベント順序
Fig. 7　Orderings of events on each object.

* 通常，型は関数的な領域を規定する意味で使われるが，ここではその概念を拡張し，時間的振舞いの領域を規定する意味で用いている[13]．

序関係 V の部分集合をなしており，型 V の副型として，次のように定義する．

[定義2] ある協調動作を構成するイベント群の全順序関係を，それらが関与する協調動作の型の副型と定義する．

2.5 型の構造化

前節で定義した型は，各オブジェクトが基本状態遷移を行うときの協調動作の枠組を与えている原始型であり，これをもとに各オブジェクトが正規表現で表される振舞いをしたときの協調動作を表す型を定義する必要がある．それをここでは型の構造化と呼び，導出された型を構造型と呼ぶ．この型の構造化は，正規表現で表された状態遷移を基本状態遷移へ分解したときの逆順に，次のように合成していく．

(1) 状態遷移からみて分岐により発生した協調動作に対応した型 $A, B, \cdots$ をもつとき，これらを合成した型はこれらの和集合と考え，$A+B+\cdots$ と表す．

(2) 型 Q を繰り返しで持つ協調動作に対応した型は，型 Q の巾集合と考え，Q^* と表す．そして，その繰り返しを内蔵ループとしてもつ親元の状態遷移に対応した型を P とすると，型 P と内蔵の型 Q^* を統合した型を $P\cdot Q^*$ と表し，前者が後者を内蔵することを示している．さらに，$Q^*;R^*$ と列記すると Q^* と R^* が逐次に行われる協調動作の型を表すものとし，$Q^*\|R^*$ とすると Q^* と R^* が並行に行われる協調動作の型を表すものとする．ただし，1つのオブジェクトが並行する協調動作に参加することはできないので，並行の記述は後述するようなシステムの振舞いを記述するときに出現する．

(3) 上記(1)または(2)を帰納的に適用していく．

以上の型の構造化の規則に従って，図4に示す各オブジェクトの振舞いの型を求めると，次のようになる．

$$O_1 : (V_1+W_1)\cdot(X_1\cdot Y_1^*)^* \tag{14}$$

$$O_2 : (V_2+W_2)\cdot(X_2\cdot Y_2^*)^* \tag{15}$$

$$O_3 : (V_3+W_3)\cdot Z_3^* \tag{16}$$

$$O_4 : (V_4+W_4)\cdot Z_4^* \tag{17}$$

ここで，V, W, X, Y, Z は，それぞれ協調動作 $F_1, \cdots, F_5$ に対応した型とし，添字を付けてそれぞれのオブジェクトに対応させた副型とする．オブジェクトの型として協調動作を表す型を対応させると，そのオブジェクトが関与する協調動作の観点からの型の宣言となり，副型にすると協調動作の中でのそのオブジェクトの振舞いを宣言することになる．

2.6 システムの型

並行オブジェクト群全体を1つのシステムとして見たとき，システムとしての振舞いの型を定義する．そのため，次の手順でシステム内で行われている協調動作を重畳し，システムとしての型とする．

(1) 各オブジェクトの型が副型で表されているときには，それを含む協調動作を表す型に置き換える．

(2) システム外で発生したイベントを受け取るオブジェクトから1つを選び，その型をシステムの型としての初期値とする．

(3) 残っているオブジェクトの中で，システムと共通の協調動作を有するオブジェクトの中から1つを選び，**図8**に示す型の合成ルールに基づき，オブジェクトのもつ型をシステムの型に併合する．

(4) システム内のオブジェクトの型がすべて併合されるまで(3)を繰り返す．

上記の手順により，図4に示す4つのオブジェクトからなるシステムの型を求めてみる．まず，各オブジェクトの副型を協調動作の型に直すと，見かけ上(14)～(17)式の添字を取り除いて，

$$(V+W)\cdot(X\cdot Y^*)^* \tag{18}$$

$$(V+W)\cdot Z^* \tag{19}$$

の2つになる．これらは排他的な協調動作 $V+W$ にお互い無関係な内蔵の協調動作を並行動作させている

2つの協調動作	統合した協調動作
A^* , B^*	$A^*\|B^*$
$A+B$, $A+C$	$A+B+C$
$A^*\|B^*$, $A^*;B^*$	$A^*;B^*$
$A^*\|B^*$, $(A\cdot B^*)^*$	$(A\cdot B^*)^*$
$A^*\|B^*$, $A^*\|C^*$,	$A^*\|B^*\|C^*$
$A^*;B^*$, $A^*;C^*$	$A^*;(B^*\|C^*)$
$(A\cdot B^*)^*$, $(A\cdot C^*)^*$	$(A\cdot(B^*\|C^*))^*$
$A^*;B^*$, $B^*;C^*$	$A^*;B^*;C^*$
$(A\cdot B^*)^*$, $(B\cdot C^*)^*$	$(A\cdot(B\cdot C^*)^*)^*$
$A^*;C^*$, $B^*;C^*$	$(A^*\|B^*);C^*$
$A^*;B^*$, $(A\cdot B^*)^*$	×
$(A+B)^*$, $A^*\|B^*$	×
$(A+B)^*$, $A^*;B^*$	×
$(A+B)^*$, $(A\cdot B^*)^*$	×

ここで，A , B , C は共通の協調動作を含まない任意の構造をもった協調動作．×は合成が許されない組合せ．

図8 協調動作の型の合成ルール
Fig. 8 Composite rules of types of cooperative behaviors.

と考えることができ，これらに図8の7番目のルールを適用すると，システムとしての型は次のようになる．

$$(V+W)\cdot((X\cdot Y^*)^*\|Z^*) \qquad (20)$$

3. 共通問題に対する応用例

本章では，いろいろなプログラミング・パラダイムを比較検討するために設定された共通問題である在庫管理システムを取り上げる[5]．この問題をオブジェクト指向モデルで設計し，オブジェクトの振舞いを前節で定義した型を用いて形式化してみる．

3.1 在庫管理システムのオブジェクト指向による設計

このシステムはいくつかの銘柄の酒をつめたコンテナの入荷と，1件当たり1銘柄の出庫依頼に応じなければならない．コンテナ入荷により，不足品リストに載っている銘柄の出庫調査をし，出庫依頼に際してはコンテナリストの在庫調査を行って依頼銘柄を出庫するが，在庫数量が不足すれば不足品リストに登録する．

この問題をオブジェクト指向で解くために，受付係にシステム管理の役目を担わせ，ほかにコンテナリストおよび不足品リストの2つのオブジェクトを設定する．受付係は待機中に受けるメッセージによって，コンテナ入荷に伴う不足品調査か，出庫依頼に伴う在庫調査を行う．**図9**にコンテナ入荷に伴う不足品調査における受付係の状態遷移を PAD 風に表す．

3.2 オブジェクトの型の導出

まず，受付係，不足品リストおよびコンテナリストを，それぞれ基本状態遷移に展開する．次に，これら基本状態遷移間での協調動作を考えると，すべての動作は受付係を中心に進められていることから，結局受付係の基本状態遷移の数だけの協調動作を設定することになる．そのうちの1例を**図10**に示す．この協調動作は，ある不足品の入荷調査を行い，これまでの不足量を解消することができたので，その不足銘柄を不足品リストから削除する場合を表している．このようにして列挙された協調動作を原始型とし，分岐に伴う排他的協調動作の型の集合を取ると，次の5つの協調動作の型に集約することができる．

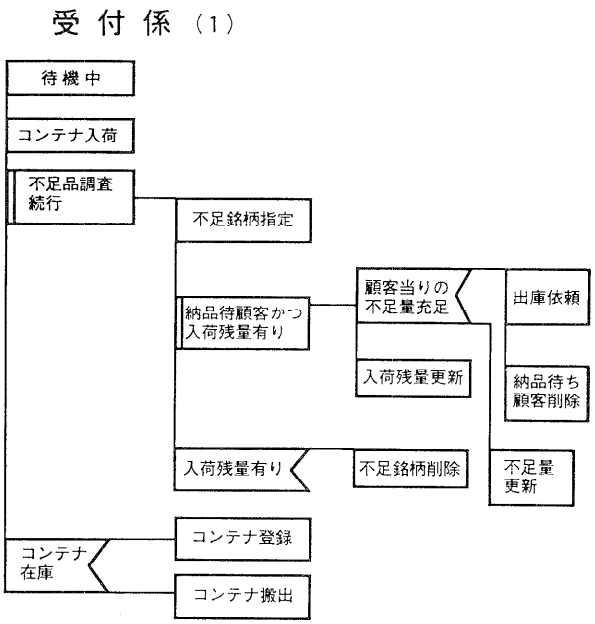


図 9　不足品調査における受付係の状態遷移
Fig. 9　The state transition of the receptionist on the investigation of the shortage of goods.

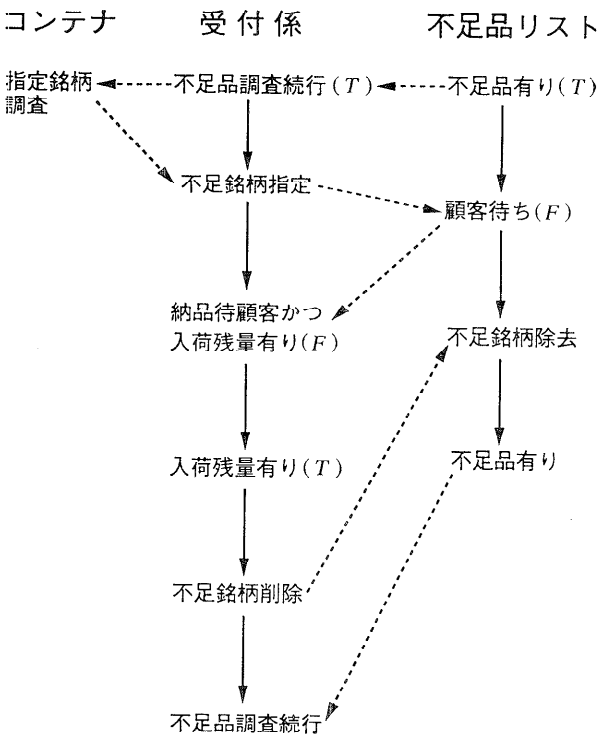


図 10　不足品調査における協調動作の1例
Fig. 10　An example of cooperative behaviors on the investigation of the shortage of goods.

$H_1=\{H_1^1, H_1^2\}$ (21)

(コンテナ入荷，登録)

$H_2=\{H_2^1, H_2^2\}$ (22)

(不足品ごとの入荷調査)

$H_3=\{H_3^1, H_3^2\}$ (23)

(納品待顧客ごとの出庫調査)

$H_4=\{H_4^1, H_4^2\}$ (24)

(顧客からの出庫依頼)

$H_5=\{H_5^1, H_5^2, H_5^3, H_5^4\}$ (25)

(依頼品の在庫調査)

ここで，上つきの添字で原始型を区別し，下つきの添字で集約した型を区別しており，ここでは各オブジェクトに副型を与えていない．

これらの基本的な協調動作を表す型をもとに，各オブジェクトの型を与えると次のようになる．

受付係：$(H_1\cdot(H_2\cdot H_3^*)^*+H_4\cdot H_5^*)^*$ (26)

不足品リスト：$(H_1\cdot(H_2\cdot H_3^*)^*+H_4)^*$ (27)

コンテナリスト：$(H_1+H_4\cdot H_5^*)^*$ (28)

この問題の場合には，システムの型は受付係の型と同じになる．これは受付係がすべての協調動作に介入していることを意味している．

3.3 協調動作に基づく型に対する考察

オブジェクトの振舞いに与えた型は，オブジェクトが意味的にもっている振舞いに関する情報を形式化したものである．これにより，静的と動的な環境で，次のようないろいろな利点を生むことができる．

1) 静的には，オブジェクト間の整合を協調動作の観点から判定することができる．オブジェクトは協調動作を行うことが前提なので，あるオブジェクトのもつ原始型に対し，他の1つ以上のオブジェクトが関与していることが必要条件である．さらに，各オブジェクトの振舞いが協調動作の副型で表されているときには，協調動作に対する各オブジェクトの関与の仕方も宣言されていることになる．これに対し，構造型は協調動作のコンテクストを与えており，各オブジェクトが守るべき協調動作の構造を与えている．**図 11** は2つのオブジェクトの構造型が整合する条件を模式的に表したものである．両者に共通する原始型のみからなる構造型は両者が関与する協調動作の構造を与えており，これがコンテキストを変えないで整合することを要求している．

2) 動的には，振舞いの型はイベントの生起順序を規定するものとなる．リアクティブ・システムのプロトタイピングの場合のように，イベントの生起順序で時間性をシミュレートしようとするときには，振舞いの型に従ってイベントを発生させればよいので，型の情報はきわめて重要である．型に基づき，任意のイベントの生起断面で各オブジェクトの状態を判定することができ，不特定オブジェクトへメッセージの放送をするような場合にも，どのオブジェクトでメッセージが受信されるべきかを判断することができる．

我々はオブジェクトの振舞いの観点から型を導出し，それによりオブジェクトおよびシステムの振舞いの仕様を記述しようとした．この基になっているのは並行オブジェクト間での協調動作であり，我々はこれをオブジェクトの振舞いの正規表現から出発して型の根拠とした．しかし，システム設計の立場からは全く逆のアプローチも考えられる．すなわち，まずシステムの振舞いの型による分析があり，これはいくつかの並行オブジェクトの協調動作で実現され，各オブジェクトはどの協調動作に関与するかが決定されていく．本論文で定義された型はメッセージの種類とそれらの呼出し順序に関する，いわば時空間領域を規定しているとも考えることもでき，特にアクティブなオブジェクトの部品化に当って，より形式的な仕様を与えることができる．これら設計過程に本論文で定義した型をどのように適用していくかは今後の課題である．

4. 関連する研究

オブジェクトが逐次的に処理される場合には，各オブジェクトでの処理はブラックボックスと見なして，入出力メッセージに基づく型解析が行われる[6),7)]．何故なら，処理中のオブジェクトを除いて他のオブジェクトは処理を中断しており，従って処理中のオブジェクトの振舞いには関知せず，ただメッセージの受理条

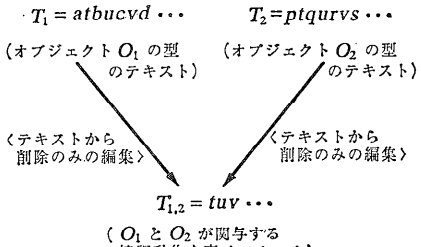


図 11 構造型の整合チェックに関する模式図
Fig. 11 A schematic diagram about the coordination check on structured types.

件および返答メッセージでのみ，お互いの接点があるからである．

しかし，オブジェクト群が並行して処理を進めていく場合には，オブジェクトはお互いが振舞いを通して相互作用を及ぼし合うことになり，各オブジェクトの振舞いをホワイトボックスと見た相互作用の解析が必要となってくる．

Eiffel では，サーバとクライアント間で一種の契約をかわし，プログラム設計することを考えており，オブジェクトの振舞いはオブジェクトの不変式で表そうとしている[8]．

Helm らは，オブジェクト間の協調と各オブジェクトの振舞いを形式化するため，各オブジェクトの振舞いに契約上の制約を課し，オブジェクトの振舞いに依存関係を定義する枠組 Contracts を導入している[9]．酒井はオブジェクトの協調動作を記述するためにコントラクトの概念を定義した[10]．そこでは，オブジェクトのライフサイクルに基づく状態遷移の過程でオブジェクト間の状態遷移，状態制約，および状態推移と事象推移をスクリプトとして表した．さらに酒井らは，オブジェクト間の協調の抽象化を進め，オブジェクトの状態遷移に基づくライフサイクル間制約を定義し，これに基づくオブジェクト群の協調動作からなるサブシステムの抽象化を行った[11]．

中島は代数的仕様記述言語 OBJ に基づく仕様記述言語 Gilo の中で，コラボレーション概念を表現するために，コラボレーションへの参加クラスの宣言，状態遷移関数の定義および各状態での機能的振舞いを示す関数の定義からなるモジュールにより，オブジェクトの集団的な振舞いを厳密に記述しようとした[12]．

これらは，オブジェクトの協調的振舞いを表現するために，メッセージの授受における制約，オブジェクトの状態遷移や事象間における制約を与える枠組の設定に関する提案である．これに対し，Nierstrasz はオブジェクトが提供するサービスを表すため，サービスタイプを提案し，そしてオブジェクトを正規プロセスに基づいてその動作を正規タイプで表現し，オブジェクトの置換性を決定するための性質を調べた[13]．原田は CCS に基づくオブジェクトの振舞いの型を定義し，グラフに基づくオブジェクト間での振舞いの整合性の解析を行った[14]．われわれは並行オブジェクトの動作が正規表現で表されるものとして，並行オブジェクト間で行われる協調動作を構造的にとらえ，各並行オブジェクトおよび並行オブジェクト群全体からなるシステムの型を定義した．この型記述により，オブジェクトおよびシステムの振舞いの仕様を簡潔に表現することができるものと考えている．さらに，これまでに提案されている協調的振舞いの枠組は，われわれが提案する振舞いの半順序関係で捉えた原始型の記述に取り込むことができる．

5. おわりに

並行オブジェクトの状態遷移が正規表現で表されるものとして，オブジェクト群の振舞いに構造的な型を定義した．これにより，オブジェクト群の静的および動的な整合性の判定が可能となる．今後，ここでの議論をつめるためにも，オブジェクトの状態遷移を正規表現で記述することの妥当性，さらにはより拡張した表現のもとでの型の導出を検討する必要があるものと考えている．

さらに，本論文で定義した型の概念を並行オブジェクトシステムの設計やオブジェクトの部品化などに有効に機能させることができると考えているが，その具体化に当っては様々な問題点を検討しなければならず，今後の検討に委ねられている．

謝辞 本研究は日本学術振興会の論博事業に基づいて，著者の一人が日本に滞在中に行われたものであり，その機会を与えてくれた武漢大学の学長を始め，同大学ソフトウェア工学国家重点研究実験室の諸氏，および貴重な討論をいただいた北海道大学工学部情報工学科言語情報工学講座の赤間清および三谷和史の2氏に感謝申し上げます．

参考文献

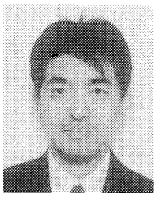
1) ランボー，J. ほか：オブジェクト指向方法論 OMT，トッパン (1992)．
2) 原田康徳，浜田 昇，渡辺慎哉，宮本衛市：多言語分散環境のための型機構，コンピュータソフトウェア，Vol. 9, No. 2, pp. 63-74 (1992)．
3) Campbell, R. H. and Habermann, A. N.: The Specification of Process Synchronization by Path Expression, *Lecture Notes in Computer Science*, Vol. 16, pp. 89-102, Springer-Verlag (1974)．
4) Bruegge, B. and Hibbard, P.: Generalized Path Expression: A High Level Debugging Mechanism, *Proc. Software Engineering Symp. on High-Level Debugging*, pp. 34-44 (1983)．
5) 二村良彦 ほか：新しいプログラミング・パラダイムによる共通問題の設計，情報処理，Vol. 26, No. 5, pp. 458-459 (1985)．

6) Agrawal, R., DeMichiel, L. G. and Lindsay, B. G.: Static Type Checking of Multi-Methods, *Proc. OOPSLA '91*, pp. 113-128 (1991).
7) Ghelli, G.: A Static Type System for Message Passing, *Proc. OOPSLA '91*, pp. 129-145 (1991).
8) Meyer, B.: Applying "Design by Contract", *COMPUTER*, Vol. 25, No. 10, pp. 40-51 (1992).
9) Helm, R., Holland, I. M. and Gangopadhyay, D.: Contracts: Specifying Behavioral Compositions in Object-Oriented Systems, *Proc. ECOOP/OOPSLA '90*, pp. 169-180 (1990).
10) 酒井博敬：オブジェクトの振舞いに関するコントラクトの設計について，情報処理学会論文誌，Vol. 33, No. 8, pp. 1052-1063 (1992).
11) 酒井博敬，堀内 一：オブジェクトの協調の抽象化について，情報処理学会研究報告，93-DBS-94, pp. 135-144 (1993).
12) 中島 震：オブジェクトの集団的振舞いの設計，情報処理学会研究報告，93-SE-95, pp. 39-46 (1993).
13) Nierstrasz, O.: Regular Types for Active Objects, *Proc. OOPSLA '93*, pp. 1-15 (1993).
14) 原田康徳：部品の接続を考慮した通信仕様の分解，日本ソフトウェア科学会第10回大会論文集，pp. 365-368 (1993).

(平成6年 6月 27日受付)
(平成6年12月 5日採録)

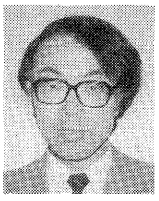
何 克清

1947年生．1970年武漢大学数学科卒業．同年より武漢大学計算機科学科でハードウェアの研究開発，1975年よりOSについての解析，評価に従事．1980年より岩手大学大学院工学研究科情報工学専攻および日立製作所ソフトウェア工場で研修．1982年武漢大学計算機科学科ソフトウェア工学研究室主任・講師，同副教授を経て，現在同大学ソフトウェア工学国家重点研究実験室主任，教授．オブジェクト指向に基づくソフトウェア開発モデル，プロセス，方法論，ツール，環境，ならびにソフトウェア開発の知識表現と利用などの研究に従事．著書に「計算機ソフトウェア工学」，「ソフトウェア工学方法」など．IEEE会員，武漢ソフトウェア工学学会理事長．



渡辺 慎哉

1959年生．1983年北海道大学工学部応用物理学科卒業．1988年同大学院工学研究科情報工学専攻博士後期課程単位取得退学．同年，北海道大学工学部助手，現在に至る．並行計算モデル，分散オブジェクトモデルなどに興味を持つ．ソフトウェア科学会会員．



宮本 衛市（正会員）

1940年生．1962年北海道大学工学部電気工学科卒業．1964年同大学院修士課程修了．同年同大学工学部電気工学科講師，同助教授を経て，1984年より情報工学科教授．工学博士．分散システム，並列オブジェクト指向モデル・設計論，プログラミング環境などの研究に従事．著書に「PASCAL—プログラミングと翻訳技法」など．電子情報通信学会，日本ソフトウェア科学会，人工知能学会，IEEE各会員．