ソフトウエア工学 17－1
（1981. 2. 12）

抽象プログラム・モデルと機能写像

谷津行穂，大場充，滝本法良，門田富史
（日本アイ・ビー・エム　製品保証）

1．はじめに

近年，ソフトウェアはその機能に対する要求の複雑さや多様性のために，構造の複雑化と大規模化が避けられない状況に陥っていると言える。また，商品としてのプログラムであるソフトウェアは，単なる処理機能面だけでなく使用する人間とのインタフェースを改善するために，互いに矛盾する「使い易く，速く」という問題や，従来のソフトウェアとの互換性についての要求も十分考慮して設計されねばならない。このような開発環境にあって，ソフトウェアの設計に関する基本的な道具である設計言語や抽象モデルは重要である。そのような設計用具として既に提案されたものに，HIPOやミルズのPDL[1]の外，有限状態機械モデルを解釈／実行するような道具も開発されている[2]。また最近では，ペトリ・ネットを基礎にいくつかの新しい仕様・設計言語が提案されている[3][4]。本小論ではファンクタと呼ばれる抽象プログラム・モデルとそのファンクタの機能を定義する機能写像（functional mapping）の概念を基礎に対象プログラムの制御構造を明確にし，見通しの良いソフトウェア・アーキテクチャを定義・記述する方法について議論する。

2．抽象プロセサ概念

情報処理システムは一般に，情報処理の論理的流れを決定し処理プロセスの操作を制御する「決定プロセス」とその決定プロセスからの要求に対する反応として具体的な情報処理機能を行なう「処理プロセス」との協働動作であると言える。この抽象化されたプロセサの考え方をプログラムに対しても適用するとアルゴリズムとデータの分離という考えが明確になる。抽象プロセサは対象データ処理機械の行動をシミュレートするアクタを発展させたものである[5]。抽象プロセサ内は，制御機能をもつ核の部分とデータ処理機能をもつデータ・フロー機械とから成っている（図1）。抽象プロセサ概念で表

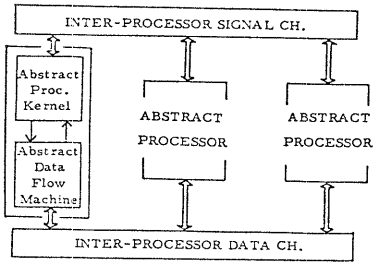


図1　抽象プロセサ概念

わされたハードウェアのもとで実行される抽象プログラムは，ファンクタ（Functor）と呼ばれる任意の機能を実行する抽象プログラム・モデルと，ゲート（Gate）と呼ばれるファンクタ間のインタフェースを定義するものから成っている。このファンクタ概念はアクタ概念を拡張するために，機能間のインタフェースであるゲートを導入している。さらに，これらの概念の表記法として近年脚光を浴びてきているペトリ・ネットを拡張した記法を用いている。セマフォア表記および選択肢表記の導入がそれである。ファンクタ間インタフェースを定義するゲートには次の3種類あり図2（a）のように記す。

- Fゲート：FORK機能を行なうゲート
- Jゲート：JOIN機能を行なうゲート
- Eゲート：可算セマフォア機能を行なうゲート

また，ファンクタには次の5種類あり

図2(b)のように記す．

- 開始ファンクタ：モデルの出発点を表現するファンクタ
- 終了ファンクタ：モデルの停止点を表現するファンクタ
- 通過ファンクタ：通常の処理を行なうファンクタ
- 通知ファンクタ：POSTやセマフォアの解放などの制御的機能を行なうファンクタ
- 選択ファンクタ：条件により異なる処理を行なうファンクタ

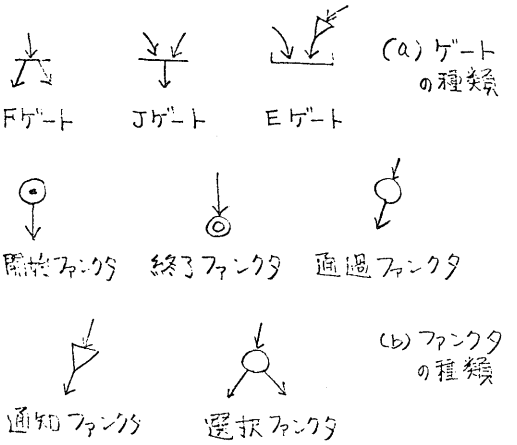


図2　ファンクタ，ゲートの表記法

ファンクタは任意の機能を実行する抽象プログラム・モデルで，1つのファンクタ核と複数個のデータ処理手順とから構成される．ファンクタ内部の制御フローは，ゲートによる起動とファンクタの構成要素であるクラスタの内部制御変数の値，および状態遷移で表現される．クラスタは複数の状態をもつpdオートマトンであり，入力と内部制御変数の値により出力を決定する．(図3)

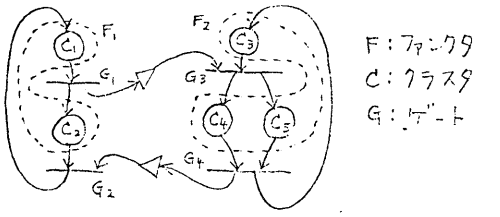
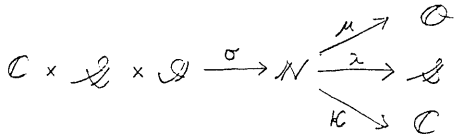


図3　ファンクタ，クラスタ，ゲート

3．機能写像

ファンクタ核はその入力，ファンクタ核自体の内部状態，および内部制御変数の値によって変化する出力を生成する．それらの関係を入力から出力へのマッピング[6]としてとらえると図4のような機能写像の構成が考えられる．



$\mathcal{I}$：入力値の集合
$\mathcal{O}$：出力値の集合
$\mathcal{S}$：状態変数値の集合
$\mathcal{C}$：内部制御変数値の集合
N：整数の有限集合

図4　機能写像の構成

そこで各クラスタのレベルへ抽象化レベルを下げていくと，クラスタ自体をファンクタと考えることができる．各クラスタのファンクタ核はpdオートマトンであり，同じように$\mathcal{I}$，$\mathcal{O}$，$\mathcal{S}$，$\mathcal{C}$の4要素で定義できる．

4．ファンクタ仕様言語

機能写像概念に基づきファンクタの制御構造を定義するファンクタ仕様言語FSL (Functor Specification Language) の概要について述べる．FSLは，それを用いて記述された抽象プログラムの仕様を翻訳して，写像σ，μ，λおよびκに対応する行列$\mathbb{\Sigma}$，$\mathbb{M}$，$\mathbb{\Lambda}$，および$\mathbb{K}$を出力するための言語である．この翻訳過程において，記述された抽象プログラムに等価な行列$\mathbb{\Sigma}$，$\mathbb{M}$，$\mathbb{\Lambda}$，お

よびKが存在するか否かが判定されるとともに，行列Cが十分かどうかが分析される．

FSLによるプログラム・モデルの仕様は以下の3つの下位仕様から構成される．

・構造仕様（structural specification）
・ゲート仕様（gate specification）
・ファンクタ仕様（functor specification）

構造仕様は，モデル内部においてファンクタとゲートがどのように組み合わされているかを記述するものである．ゲート仕様は，モデル内のゲートについて，その機能とクラスタを起動する条件を定義するものである．ファンクタ仕様はファンクタの内部に関して以下の2点を定義する．

・ファンクタを構成するクラスタとゲートの結合関係
・クラスタ別のファンクタ核構造

FSLの全体構造は図5に示すとうりである．

```
begin specification : EXAMPLE ;
  begin structure ;
    functor   F1 ( P1, P2 ; P3, P4 ) ;
              F2 (Q1,Q2,Q3,Q4;Q5,Q6);
    gate      G1 ( P4 ; P1, G3 ) ;
              G2 ( P2, G4 ; P3 ) ;
              G3 (G6, G1 ; Q1, Q2 ) ;
              G4 (G3, Q4 ; Q5, G2 ) ;
  end ;
  begin gate : G1 ;
    type  F-gate ;
    input   P4 ;
    operation   V-op ;
    output   P1 , G3 ;
  end ;
  begin gate : G2 ;
    ⋮

  begin functor : F1 ;
    declare  cluster : C1 ( P3 ; P4 ) ;
             cluster : C2 ( P1 ; P2 ) ;
    begin cluster : C1 ;
      ⋮
      <クラスタ C1 の定義>
      ⋮
    end ;
    ⋮
  end ;
  begin functor : F2 ;
    ⋮
  end ;
end ;
```

図5　ファンクタ仕様言語構造

ファンクタ核は図5のようにクラスタ別に定義される．各クラスタのファンクタ核はpdオートマトンとして以下の4つの要素で記述される．

・状態の遷移
・入力
・出力
・内部制御変数の条件と変更

入力や出力には，ファンクタ外部への外部入出力と，ファンクタ内部のデータ処理手続きとの内部入出力とがある．ある状態から他の状態への遷移とその条件および内部制御変数値の変更は，図6に示される5つのステートメントによって記述される．

```
begin
  prestate   <状態遷移が生じる直前の
             状態>;
  input      <状態遷移のきっかけとな
             る入力>;
  control    <内部制御変数の条件と状
             態遷移後の内部制御変数の
             変更についての記述>;
  output     <状態遷移の結果生じる出
             力>;
  poststate  <状態遷移後に到達する状
             態>;
```

```
end ;
```

図6　FSLにおける状態遷移の記述

5．まとめ

ファンクタに関する考察およびその表記法等について述べたが，プログラムの制御構造を明示的に表現するためにはかなり強力な表記法であると思われる．また，ファンクタ概念を記述する言語として文脈自由言語の性質を備えたFSLの概要について簡単に述べた．現在，われわれは上述したFSLを用いた実験を企画すであり，この実験によりFSL自身の仕様を改善することが今後の課題である．

参考文献

1) Teichroew D., et al., PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems, IEEE Trans. on Software Eng., Vol. SE-3, No. 1, 1977.

2) Berthaud M., et al., Toward a Formal Language for Functional Specifications, Proc. of the IBM Interdivisional Sympo. on Communications, Networks and Distributed Function, 1977.

3) Mekly L.J., et al., Software Design Representation Using Abstract Process Networks, IEEE Trans. on Software Eng., Vol. SE-6, No. 5, 1980.

4) Trattnig W., et al., EDDA, A Very-High-Level Programming and Specification Language in the Style of SADT, Proc. of COMPSAC 80, 1980.

5) Hewit C., et al., Laws for Communicating Parallel Processes, Proc. of IFIP CONGRESS, 1977.

6) Ohba M., et al., Architecture Kernel: Higher-Level Program Specification, Proc. of COMPSAC 80, 1980.