1Q−7

Griesの導出法によるObject−Z仕様からのプログラム導出

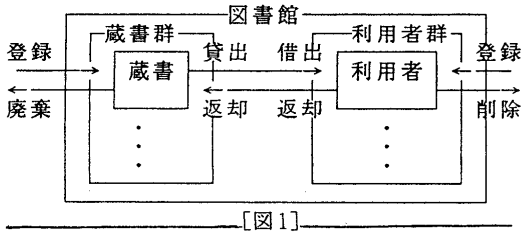
宮崎比呂志, 染谷誠, 山崎誠一, 谷津弘一
情報処理振興事業協会, 日本ユニシス, NTT ソフトウェア研究所, 日本ユニシス

1. はじめに

Object-Z(VDM, Z)[1],[2],[3]などの集合に基づいた形式的仕様記述言語は，概念的なモデル で仕様化するシステムの状態空間を表現し，その状態空間に作用するオペレーション を定義するものである．これらは概念的な設計が可能なものである．その反面，これらの仕様記述言語は"looseness"を持つ．たとえば，インプリメントのデータ 構造を規定しない，論理和などで記述された仕様の実行順序が非決定的であるなどがある．このような問題により，これらの仕様は，実行不可なものである．本稿では，これらの問題点を解決するための1 つのアプローチ を提案する．これを検証をするために仕様記述/ 導出実験を行った．実験例として，"ソフトウェアの仕様記述と設計に関する第 4回国際ワークショップ"の課題の1 つである図書館問題を取り上げる．

2. 図書館のクラス 構成

Object-Zの特性に基づき，図書館を構成するクラス を考える．図書館を考えた場合，オブジェクト指向でいう"オブジェクト"として" 利用者" と" 蔵書"に着目することは自然であると考える．従って，これらのオブジェクトが属する"クラス" を定義する．そして，利用者全体，蔵書全体を表すクラス として" 利用者群"," 蔵書群" を定義する．これらの" 利用者群"," 蔵書群" を管理するのが,"図書館"クラスとなる．この関係を図1 に示す．



[図1]

3. 仕様(紙面の都合上，貸出だけを仕様化する

蔵書

蔵書
ト(現状，貸出先，貸出日，貸出，返却)

book: 本
category: 分野
status: 蔵書状態
borrower: 利用者ID
bordate:日付

status = "在庫中" ⇔borrower = ⊥
∧bordate = ⊥

貸出
Δ(status,borrower,bordate)
uc?:利用者ID
d?: 日付

status = "在庫中"
status' = "貸出中"
borrower' = uc?
bordate' = d?

蔵書群

蔵書群
ト(登録済，該当書，登録，廃棄，貸出，返却)

copies: 蔵書ID ⇸ 蔵書

貸出
Δ(copies)
cc?:蔵書ID
uc?:利用者ID
d?: 日付

cc? ∈ dom(copies)
let c == copies(cc?)・∃c': 蔵書 |
c.貸出・copies' = copies⊕ {cc? ↦ c'

利用者

利用者
ト(借出中，借出本無し，制限未満，借出，返却
max:N1

person: 人
licence:利用者区分
borrec: 蔵書ID⇸日付

借出
Δ(borrec)
cc?:蔵書ID
d?: 日付

#borrec < max
¬cc? ∈ dom(borrec)
borrec' = borrec∪ {cc? ↦ d? }

Program Derivation from Object-Z Specification using Gries Method

Hiroshi Miyazaki, Makoto Someya, Seiichi Yamazaki Kouichi Yatsu
IPA , Nihon Unisys, Ltd, NTT Nihon Unisys, Ltd

利用者群

```
利用者群
├( 登録済,該当者,登録,削除,借出,返却)
┌──────────────────────────
│ users:利用者ID↠利用者
└──────────────────────────
借出
Δ(users)
uc?:利用者ID
cc?:蔵書ID
d?: 日付
uc? ∈ dom(users)
let u == users(uc?) ・∃u': 利用者 |
  u.借出・users' = users⊕{uc? ↦u'}
```

図書館

```
図書館
├( 蔵書登録済, 利用者登録済, 該当蔵書,
   該当利用者, 蔵書登録, 利用者登録,
   蔵書廃棄, 利用者削除, 貸出, 返却)
┌──────────────────────────
│ cfile:蔵書群
│ ufile:利用者群
└──────────────────────────
∀cc: 蔵書ID;uc:利用者ID・
 (∃ufile.該当者 | uc? = uc
・(∃u!.借出中 | cc? = cc・v! = true)) ⇔
 (∃cfile.該当書 | cc? = cc
・(∃c!.貸出先・uc! = uc))
Init ≙ cfile.Init ∧ ufile.Init
蔵書登録 ≙ cfile.登録
利用者登録 ≙ ufile.登録
蔵書廃棄 ≙ cfile.廃棄
利用者削除 ≙ ufile.削除
貸出 ≙ cfile.貸出∧ufile.借出
```

4.ﾌﾟﾛｸﾞﾗﾑ の導出

ﾌﾟﾛｸﾞﾗﾑ を導出するために，前述のObject-Zで記述された仕様を同値なGries の仕様に変換して,ﾌﾟﾛｸﾞﾗﾑを導出する．紙面の都合上，図書館，蔵書群，蔵書，利用者群，利用者ｸﾗｽ はそれぞれ下位のﾓｼﾞｭｰﾙ 呼出し程度の処理しか行わないので，ｵﾍﾟﾚｰﾀ 群のｵｰﾊﾞﾗｲﾄﾞ(⊕) だけについて述べる．具体的なﾃﾞｰﾀとして配列を用いる．(Griesでは配列を用いるのが，最適である.)

#mapr →

ident	values
・・・	・・・
・・・	・・・
・・・	・・・

mapr

⊕は定義は(*ma*,*mb* :map)

ma⊕*mb* ≙

{*d* ↦(if *d* ∈dom *mb* then *mb*(*d*)
else *ma*(*d*) | *d* ∈(dom *ma* ∪dom *mb*) }

```
proc override(mapr, id, val)
  value result  mapr:ID ↠VARIABLES
  value         id:ID
  value         val:VARIABLES
pre  mapr = M ∧∃i:0 ≦i <#mapr:
                mapr(i).ident = id
post #mapr = M.#mapr∧∀i:0 ≦i <#mapr:(
(mapr(i).ident = id ⇒
     mapr(i).values = val)∧
(mapr(i).ident ≠ id ⇒
     mapr(i).ident = M.mapr(i).ident∧
     mapr(i).values = M.mapr(i).values))
```

この仕様を基に不変条件を求め，この不変条件を満たすことを証明することによって以下のﾌﾟﾛｸﾞﾗﾑ が得られる．

```
1:= 0;
do 1  ≠ #mapr
  if mapr(i).ident = id →
     mapr(i).values:= val;
  □ mapr(i).ident  ≠ id →
     skip
  fi (
  1:= 1 + 1 ;a
od
```

4.まとめ

Object-Zの仕様の"looseness" を解決するための1 つの方法が提示できたと考える．ただしここで,用いたﾃﾞｰﾀ 構造は配列であり，現実的なｲﾝﾌﾟﾘﾒﾝﾄ 方法であるとはいえない．従って，各種のﾃﾞｰﾀ 構造に対してこのｱﾌﾟﾛｰﾁ が可能であることを示す必要がある．

5.参考文献

[1]D.Carrington, D.Duke, R.Duke, P.King G.Rose and G.Smith Object-Z:An Object-Oriented Extension to Z, In formal Description Techniques(FORTE'89) North Holland, 1990

[2]J.M.Spivey, The Z Notation: A Refernce Manual, Prentice Hall, 1989

[3]C.B.Jones, Systematic Software Development using VDM Second Edition, Prentice Hall, 1990.

[4]E.W.Dijkstra. A Discipline of Programming. Prentice Hall, Englewood Cliffs, 1976 (邦訳: 浦　昭二，土居範久，原田賢一共訳「ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ 原論」ｻｲｴﾝｽ 社(1983)).

[5]D.Gries, The Science of Proramming, Springer-Verlag, 1981.
筧捷彦訳,ﾌﾟﾛｸﾞﾗﾐﾝｸﾞの科学，培風館，1991.