組み込みソフトへの数理的アプローチ

第3回

論理式のテスト
――数理的アプローチの基礎トレーニング1

藤倉 俊幸

1 形式的に表現することのメリット

● 論理式や数式を使った表現＝形式的な表現

数理的なアプローチによってソフトウェアの品質を確保する方法について，いろいろと考えていくことがこの連載の目的である．第1回と，第2回は，要求からコーディングまでの流れをざっと見た．それは，

「要求仕様から作った前条件Aの下で，プログラムの後条件または不変条件Bが設計仕様Dを満足すること，その設計仕様が要求仕様Rを満足することを証明すること」

である．つまり，BならばD，DならばRということを証明できれば，プログラムの正しさを示したことになる．

前条件A→(後条件B→(設計仕様D→要求仕様R))

これが全体の流れである．そして，この流れの根底にあるものは，A，B，D，Rを形式的に表現するということである．

形式的に表現すると書いたが，具体的には論理式や数式を利用するということである．実は数式だけでなく，プログラミング言語で書くということも形式的な表現の一種である．しかし，すでに見たように∀(すべて)やヨ(存在する)などの記号を使用した抽象度の高い表現が必要になるため，残念ながらプログラミング言語で仕様を表現することは難しい．けれどもアルゴリズムとそこで使用するデータ構造の形式的表現に限定すれば，プログラム言語でもある程度の内容を表現することが可能である．

● 意思疎通のためにも使える

抽象度の高い記号を使って考えたり，テスト・ケースを作ったりできないと，上述の形式的表現を証明するところまで到達できない．しかし，いきなり証明を目指すのは，かなり無理がある．実は，証明するところまで到達できなくても，表現手段として習得するだけでも十分なメリットがある．

以前，米国人としごとをしたときのことだ．∀やヨを使ってホワイト・ボードにプログラムの前提条件などを書く人がいて，仕様の認識などがたいへんスムーズだった経験がある．表にまとめるとか，図にするとかと同様の，一種の企業文化や風土のようなものとして抽象度の高い形式的表現を定着させることが，プログラム開発現場には必要である．この形式的な表現の基本は論理式だと思われる．プログラミング言語レベルの論理式から∀やヨを使った論理式まで守備範囲を広げると，自然言語を超えた万国共通の「考えるための道具」を手に入れることができる．

そのためには，これらの表現に慣れることが第一歩になる．その手段として，自分で論理式を書いて，書いたものが意図したとおりであるかどうかを確認する手段を確立しよう．証明に進む前には，論理式の意味を理解して，その論理式がどのようなときに真になり，どのようなときに偽になるのかわかることが必要である．これができるようになれば，テスト・ケースも作れるようになる．

実際にやってみよう．たとえば前回(2006年10月号)，「整数を要素とする配列Aの中で最大の要素」を見つけるプログラムの証明を扱った際，いちばん大きい値をもった要素のインデックスをxとして，この条件を形式的に

$$B0 = \{\ 1 <= x <= n,\ 1 <= \forall\, j <= n,\ A[x] >= A[j]\ \}$$

と表した．本当にこれで良いのか，確認しようということである．

2 テスト環境を構築する

● 形式的記述・検証ツールVDM++

この連載を執筆するにあたり，なるべく無料で使えるツールを紹介して実際に利用する，それも設定が簡単でWindows上でダブルクリックするだけで使えるツールを利用する，というポリシを設定している[注1]．

そこで今回はVDMを使用してみる．VDMにはVDM-SLとVDM++の二つがある．論理式の基礎トレーニングに利用するだけなら，どちらでも同じである．VDM++はクラスを作らなければならないが，VDM-SLのほうはそのようなことはないので，すぐ本題に入れる．けれども，VDM++のほうが頻繁にバージョンアップされていて，情報量が多いらしい．VDMの非商用版はCSKのWebサイト[注2]からダウンロードすることができるが，ユーザ登録が必要だ．名前を知られては困る人は，

注1：Linuxには捨てがたい形式手法ツールが多いのだが，今のところはWindowsで話を進める．

注2：http://www.vdmtools.jp/

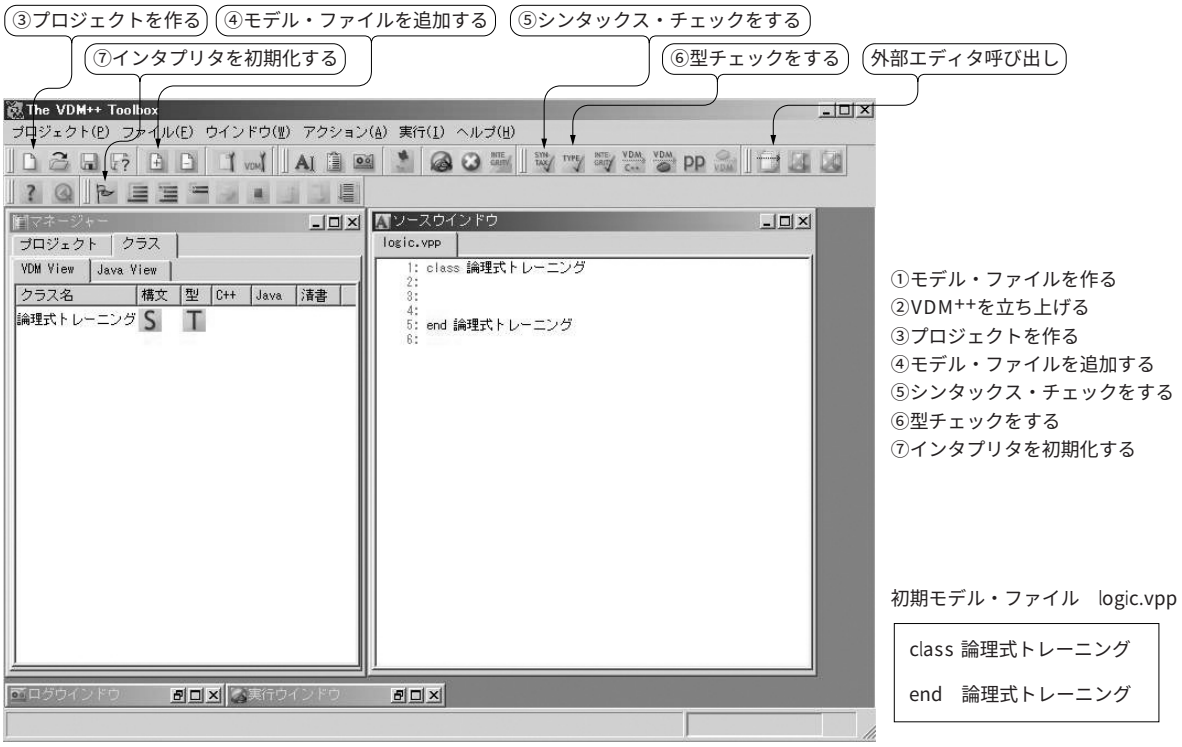

図1
VDM++ の実行準備

参考文献(1)を購入するとCDが付いてくる．CDに入っているのはVDM Tools Liteというものだが，ほとんど製品版のVDM-SLと同じ仕様にしたと作成に携わった技術者が語っていた．これらのどれを選んでも，インストール・プログラム(setup*.exe)をダブルクリックすればインストールできる．

たぶん，クラスを使用できたほうがこの先便利なことがあると思われるので，ここではとりあえずVDM++を使用することにする．インストールが終了したら，**図1**の手順で実行準備を整える．VDM++はクラスが必要なので，テキスト・ファイルにクラス名だけの初期モデル・ファイルを作り，型チェックまで進むと**図1**の状態になる．ここまでは，ボタンをマウスでクリックするだけである．これで空っぽのクラスができたので，後はこのクラスの中にテスト環境を作っていくだけだ．作っていく過程で，モデル・ファイル編集やシンタックス・チェック，型チェック，コマンドの再実行のループを何度も回すことになる．このとき，使用するテキスト・ファイルを外部エディタとして登録しておくとか，[↑]キーのヒストリ機能などに慣れるなどしておくといいだろう．そうすれば，ループが佳境に入ったときに編集したモデル・ファイルをロードし忘れるなど，つまらないことで問題が生じることがなくなる．また，VDM++のソース・ウィンドウは表示するだけでは編集ができないので，注意が必要だ．

● VDM++を使ってテストしてみる

前回の論理式B0をテストするために，モデル・ファイルを作成する(**図2**)．まず，テスト用の配列をもつ必要があるので，インスタンス変数として整数の配列A[注3]を定義する．次に，この配列に値を設定するための操作`dataSet()`と，B0を評価するための操作`maxCheck()`を定義する．B0のnは配列Aのサイズなので，具体的なAが与えられないと決まらない．VDMでは，`inds`という演算子で配列のインデックス集合を作ることができる．これを利用すれば，Aが与えられたら`inds A`とすることで自然数の集合$\{1, 2, \ldots, n\}$を得ることができる．

$1 <= x <= n$は，xがこの集合の要素であるという条件なので，(`x in set inds A`)と書くことができる．`in set`もVDMの演算子で，`x`が集合`inds A`の要素であれば真を返す．**図2**ではこの方法を使って`maxCheck()`の事前条件とした．B0に忠実にテストしたければ，インスタンス変数`n`を定義し，`dataSet()`でAが確定した際にAの長さを設定しておけばよい．配列の長さは，`len A`によってとることができる．C言語で記述するように$1 <= x$と$x <= n$の論理積で指定することもできるし，集合を使って`x in set {1,..., n}`で指定することもできる．配列はかぎカッコ[ ]で，集合は中カッコ{ }で表す．事前条件`pre`を使用した場合，事前条件違反があると，

```
>> print s1.maxCheck(5)
C:/VDM/logic.vpp, l. 21, c. 5:
  Run-Time Error 58: The pre-condition
   evaluated to false
```

のように表示される．`print`は，実行ウィンドウで実行するた

注3：VDMではseq of int，列型と呼んでいる．