VDM-SL 記述の関数適用列の可視化に関する考察

2013SE095 小松 伸
指導教員：張 漢明

1 はじめに

高い信頼性が求められるソフトウェア開発において，形式手法の有効性の事例が報告されている．開発文書に形式仕様言語を導入して設計文書の信頼性を向上させることが，ソフトウェア全体の信頼性向上に寄与する [1]．形式手法は数学の概念に基づき厳密な記述であるので，曖昧さがなくなる．仕様は対象システムの妥当性確認及び設計プログラムの正当性検証の基準として重要である．本研究で用いる形式仕様言語は，VDM-SL であり，VDM-SL を用いて機能と振る舞いを記述するスタイルを提示する．

本研究では，テスティングにより仕様記述の妥当性確認を行うことを想定する．代入や繰り返しの記述がない宣言的で簡潔な記述である関数スタイルで仕様を記述する．仕様記述を書いたときにプログラムが正しく動作しなかった場合，仕様記述が間違っていることを発見しデバッグする必要がある．しかし，仕様を記述する時に関数適用列に着目すると，関数の適用の連続によるシナリオ実行時に評価結果が表示されるまでの途中計算過程を見ることができない．デバッグする際に途中計算過程でどのような間違いが起きているかを振り返ることができず，仕様記述のデバッグが困難になる．本研究の目的は，関数スタイルの VDM-SL 記述のデバッグを支援するための関数適用列の可視化の方法を提案することである．

2 背景技術

2.1 形式手法

ソフトウェアの信頼性や安全性に対する関心が高まり，形式手法が注目を集めている．形式手法とは，仕様に対する実現の正しさを検証で確認することである．形式仕様言語は，システム，特に，ソフトウェアの開発にあたって，数理論理学に基づく科学的な裏づけを持つ．明確で厳密な意味を持つ言語を用いて設計対象を表現することにより，設計記述の正しさを順序立って示すことが可能になる [2].

2.2 VDM(VDM-SL)

形式仕様言語のことで，形式手法の発端は VDM にあると言われている．VDM-SL を用いる理由として，日本語の識別子が利用可能であり，実績のあるフリーのツール，日本語を含めたマニュアルと参考文献の存在があげられる．VDM-SL は，手続き的な記述に比べて宣言的な記述の抽象度が高いので，問題の本質に注力することができる．

2.3 関数スタイルの VDM-SL 機能仕様

関数スタイルとは，システムの状態の変化と入出力を結ぶ操作を，関数で定義したものである．操作の型を以下に示す．

```
functions
  操作名：入力型 * 状態出力型 -> 状態出力型
  操作名（入力， 入力前状態） == 入力後状態の定義

状態出力型 = 状態型 * 出力型
```

関数を用いて操作を記述するために，引数で操作前の状態をもらい，操作後の状態を返している．

3 研究のアプローチ

3.1 VDMPad の問題点

本研究では VDMPad の可視化機能を利用する．問題点を以下に示す．

- 評価したい値の評価結果の最終的な状態が得られ，途中の状態遷移がわからない．
- 仕様が大規模，複雑化に伴い内部構造も大規模，複雑化する．

3.2 基本的なアイデア

関数適用の手続き的な実行

```
操作１ -> 操作２ -> 操作３ -> 操作４
```

以上の操作において，関数適用の場合に以下の式となる。

```
操作４（操作３（操作２（操作１）））
```

VDM-SL において関数適用の連続だと煩雑であるのでわかりにくい．

```
[操作１, 操作２, 操作３, 操作４]
```

よって，以上のように記述を手続き的にする．

状態に着目した関数適用列のトレース化

ある状態における関数を再帰的に定義することにより，操作ごとの状態を記述する．

4 関数適用列の可視化

関数適用を VDM-SL において表現するために以下の関数を定義する．

```
types
  出力型 = 状態型 -> 状態型;
  入力型 = seq of (出力型);

functions
  操作列を実行:入力型 * 状態型 -> 状態型
  操作列を実行(入力, 入力前状態) ==
  if 入力 = []
  then 入力前状態
  else 操作列を実行(tl 入力, (hd 入力)(入力前状態));
```

ある操作列の初期状態から操作後の状態を示している．

このように記述することで関数適用の連続のわずらわしさがなくなる．

操作列の状態列のトレースを記述するための関数を以下に定義する．

```
操作列の状態列:入力型 * 状態型 -> seq of 状態型
操作列の状態列(入力, 入力前状態) ==
 if 入力 = []
 then []
 else let s = (hd 入力)(入力前状態)
       in [s] ^ 操作列の状態列(tl 入力, s);
```

ここである状態に着目したい場合に，操作列の状態列の関数から一部変えることにより，着目した部分のトレースを記述することができる．以下にその関数を定義する．

```
操作列の○○列:入力型 * 状態型 -> seq of 状態型
操作列の○○列(入力, 入力前状態) ==
 if 入力 = []
 then []
 else let s = (hd 入力)(入力前状態)
       in [s.○○] ^ 操作列の○○列(tl 入力, s);
```

以上の関数により与えられたトレースを VDMPad の可視化機能を利用して可視化する．

5 考察

5.1 事例：自動販売機

自動販売機を用いた操作列の例を以下に示す．

```
貨幣を投入操作(100),
貨幣を投入操作(100),
販売ボタン押下操作(販売ボタン_0),
販売ボタン押下操作(販売ボタン_0)
```

操作列の状態列を実行すると，行った操作ごとの自動販売機全体の状態遷移を表した図が得られるようになる．さらに詳細化したいので，操作列の在庫列に着目した例を以下に示す．評価結果は以下となる．

```
[{mk_token("商品_0") |-> 5,
 mk_token("商品_1") |-> 5,
 mk_token("商品_2") |-> 5},
 {mk_token("商品_0") |-> 5,
 mk_token("商品_1") |-> 5,
 mk_token("商品_2") |-> 5},
 {mk_token("商品_0") |-> 4,
 mk_token("商品_1") |-> 5,
 mk_token("商品_2") |-> 5},
 {mk_token("商品_0") |-> 3,
 mk_token("商品_1") |-> 5,
 mk_token("商品_2") |-> 5}]
```

しかし，在庫に着目したときの評価結果の内部構造のトレースの結果だけではわかりにくい．これを可視化した場合に，図 1 のようになる．

以上から，在庫のみの推移が図で示せるようになり，商品_0 が五個から三個に減っていることが容易にわかる．

操作列の出力列に着目した例を示す．評価結果は以下となる．

```
[nil, nil,
mk_出力型(mk_token("商品_0"), nil),
mk_出力型(mk_token("商品_0"), 0)]
```

図 1 在庫列のトレース

これも在庫列と同様に図 2 のように可視化することで見やすくなる．

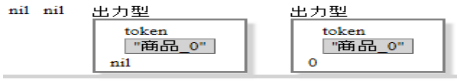

図 2 出力列のトレース

5.2 可視化の検討

操作列，操作列の状態列の内部構造は自動販売機全体の状態の結果であるので内部構造が大規模，複雑化していたが，一つの状態に着目することで自動販売機のある一つの操作の状態の内部構造をより詳細化することができた．内部構造の結果だとまだわかりにくい部分があったので，本研究では関数適用列で着目した状態のトレースを図で示せるようになった．ある一つの状態だけに着目することができるようになり，その仕様の状態の何回目の操作でどこで何が間違っていたかを詳細的に見ることができるようになり，デバッグの支援につなげることができた．

しかし，その数字が何を表しているかやどこが何に対応しているかなど場所の対応がわかりにくく見え方に問題がある．したがってその改善策として，順番や名称を示せるようになること，一つだけでなく着目したい状態を複数選択できるようにすることや一つの商品だけに着目することがあげられる．また，現状では関数を一つ一つ書かなければならないので，着目したい状態を指定したらそれに対応する関数を自動生成できるようになれば楽に関数を扱うことができると考えられる．見え方を工夫することで可視化の支援に繋がり，デバッグの支援にも繋がると考えられる．

6 おわりに

本研究では関数スタイルの VDM-SL 記述のデバッグを支援するための関数適用列の可視化の方法を提案し，仕様の妥当性を確認する手段を確立することができた．今後の課題としては，関数適用列の可視化時の見え方の工夫をすることである．

参考文献

[1] 寺西祐斗, 張漢明, 沢田篤史,” 自動販売機システム開発文書への形式手法記述の適用”,” 情報理学会研究報告”,vol.2016-SE-191,No.12,2016 年 3 月

[2] 中島震,” ソフトウェア工学の道具としての形式手法”,”National Institute of Informatics”,2007 年 7 月