運賃計算システム VDM++ 仕様

佐原　伸

法政大学大学院
情報科学研究科

概要

レコードの集合と関数を使った運賃計算の要求仕様記述例である。
型やインスタンス変数の不変条件、事後条件、事前条件、簡易回帰テストの例も含んでいる。
組み合わせテスト機能は、今回の入門コースでは説明はしないが、本モデルのテストには使用した。

目次

1 運賃表クラス

要求辞書レベルの運賃表と、関連する機能を表す。

この部分は TeX ドキュメントであり、ツールの清書機能を使って、VDM++ ソースと一体化したドキュメントを清書し、PDF ファイルとして生成することができる。

```
class
運賃表
types
  public 駅 = seq of char
  inv w 駅 == w 駅 <> "";
  public 運賃 = nat;
public
  運賃レコード :: f 駅 1 : 駅
                  f 駅 2 : 駅
                  f 運賃 : 運賃
  inv w 運賃レコード ==
      w 運賃レコード.f 駅 1 <> w 運賃レコード.f 駅 2;
  public 運賃表集合 = set of 運賃レコード
  inv w 運賃表集合 ==
      w 運賃表集合 = {} or
       forall w 運賃レコード 1, w 運賃レコード 2 in set w 運賃表集合 &
           w 運賃レコード 1.f 駅 1 = w 運賃レコード 2.f 駅 1 and
           w 運賃レコード 1.f 駅 2 = w 運賃レコード 2.f 駅 2 =>
           w 運賃レコード 1.f 運賃 = w 運賃レコード 2.f 運賃
```

1.1 運賃表を検索する

運賃表集合に存在するか判定する。

```
functions
protected
  運賃表を検索する : 運賃表集合 * 駅 * 駅 -> 運賃レコード
  運賃表を検索する (a 運賃表集合, a 駅 1, a 駅 2) ==
    let  w 運賃レコード in set a 運賃表集合 be st
              {a 駅 1, a 駅 2} = {w 運賃レコード.f 駅 1, w 運賃レコード.f 駅 2} in
    w 運賃レコード
  pre  運賃表集合に存在する (a 運賃表集合, a 駅 1, a 駅 2)
```

```
  post  exists1 w 運賃レコード in set a 運賃表集合 &
          {a 駅 1, a 駅 2} = {w 運賃レコード.f 駅 1, w 運賃レコード.f 駅 2} and
          RESULT = w 運賃レコード ;
```

1.2 運賃表集合に存在する

運賃表集合に存在するか判定する。

```
protected
  運賃表集合に存在する : 運賃表集合 * 駅 * 駅 -> bool
  運賃表集合に存在する (a 運賃表集合, a 駅 1, a 駅 2) ==
    {a 駅 1, a 駅 2} in set {{e.f 駅 1, e.f 駅 2} | e in set a 運賃表集合 }
end
```

運賃表

Test Suite : vdm.tc

Class : 運賃表

Name	#Calls	Coverage
運賃表 ‘運賃表を検索する	3	√
運賃表 ‘運賃表集合に存在する	7	√
Total Coverage		**100%**

2 運賃計算クラス

要求仕様レベルの運賃計算仕様を表す。

```
class
運賃計算 is subclass of 運賃表
```

2.1 インスタンス変数

s 運賃表集合は運賃表のデータを保持するが、テストケースで運賃表のデータを与えるため、初期値は空集合にした。

```
instance variables
  s 運賃表集合 : 運賃表集合 := {};
```

2.2 運賃計算構成子

運賃計算オブジェクトの構成子。

```
operations
public
  運賃計算 : 運賃表集合 ==> 運賃計算
  運賃計算 (a 運賃表集合) ==
    s 運賃表集合 := a 運賃表集合;
```

2.3 運賃を得る

2 駅間の運賃を計算する。

```
public
  運賃を得る : 駅 * 駅 ==> 運賃
  運賃を得る (a 駅 1, a 駅 2) ==
    if not 運賃表集合に存在する (s 運賃表集合, a 駅 1, a 駅 2)
    then exit <運賃表に存在しない>
    else let w 運賃レコード = 運賃表を検索する (s 運賃表集合, a 駅 1, a 駅 2) in
         return w 運賃レコード.f 運賃
```

```
  post  exists1 w 運賃レコード in set s 運賃表集合 &
          {a 駅 1, a 駅 2} = {w 運賃レコード.f 駅 1, w 運賃レコード.f 駅 2} and
          RESULT = w 運賃レコード.f 運賃
end
```

運賃計算

Test Suite : vdm.tc

Class : 運賃計算

Name	#Calls	Coverage
運賃計算 ‘運賃計算	4	√
運賃計算 ‘運賃を得る	4	√
Total Coverage		**100%**

3 Test クラス

運賃計算検証用の単純な回帰テストケースである。
実用システムでは VDM++ 用の回帰テストライブラリを使用するが、入門段階では説明しない。
エラーケースは、まだ考慮していない。

```
class
Test is subclass of 運賃計算
values

  v 運賃レコード集合 : 運賃表集合 = {
                                    mk_運賃レコード ("東京","品川",220),
                                    mk_運賃レコード ("東京","新宿",180),
                                    mk_運賃レコード ("新宿","品川",190)}
operations
public
  run : () ==> seq of bool
  run () ==
    return [t1 (),t2 (),t3 (),t4 ()];
public
  t1 : () ==> bool
  t1 () ==
    (   let w 運賃計算 = new 運賃計算 (v 運賃レコード集合) in
        return w 運賃計算.運賃を得る ("東京","品川") = 220
    ) ;
public
  t2 : () ==> bool
  t2 () ==
    (   let w 運賃計算 = new 運賃計算 (v 運賃レコード集合) in
        return w 運賃計算.運賃を得る ("東京","新宿") = 180
    ) ;
public
  t3 : () ==> bool
  t3 () ==
    (   let w 運賃計算 = new 運賃計算 (v 運賃レコード集合) in
        return w 運賃計算.運賃を得る ("新宿","品川") = 190
    ) ;
public
```

```
  t4 : () ==> bool
  t4() ==
    (   let w運賃計算 = new 運賃計算(v運賃レコード集合) in
        trap <運賃表に存在しない> with  return truein
        return w運賃計算.運賃を得る("ソウル","聆山") = 190
    );
public
  Tr1 : 駅 * 駅 ==> bool
  Tr1(a駅1,a駅2) ==
    (   let w運賃計算 = new 運賃計算(v運賃レコード集合) in
        trap <運賃表に存在しない> with  return truein
        return w運賃計算.運賃を得る(a駅1,a駅2) = 190
    )
end
Test
```

Test Suite : vdm.tc

Class : Test

Name	#Calls	Coverage
Test‘t1	1	√
Test‘t2	1	√
Test‘t3	1	√
Test‘t4	1	85%
Test‘Tr1	0	0%
Test‘run	1	√
Total Coverage		**76%**

4 組み合わせテストケース UseFare0

運賃計算検証用の組み合わせテストである。
組み合わせテスト自体は、まだ入門コースであるため説明しない。

```
class
UseFare0
values

  v 運賃レコード集合 : 運賃表 ‘運賃表集合 = {
                                        mk_運賃表 ‘運賃レコード ("東京", "品川", 220),
                                        mk_運賃表 ‘運賃レコード ("東京", "新宿", 180),
                                        mk_運賃表 ‘運賃レコード ("新宿", "品川", 190)}
instance variables
  sTest : Test := new Test();
  s 運賃計算 : 運賃計算 := new 運賃計算 (v 運賃レコード集合);

traces
T0 :
  sTest.run ()
  ;
; T1 :
  let w 駅 1 in set {"東京", "品川", "新宿", "ソウル"} in
  let w 駅 2 in set {"東京", "品川", "新宿", "武蔵境", "昭山", ""} in
  s 運賃計算.運賃を得る (w 駅 1, w 駅 2)
  ;
; T2 :
  let w 駅 1 in set {"東京", "品川", "新宿", "ソウル"} in
  let w 駅 2 in set {"東京", "品川", "新宿", "武蔵境", "昭山", ""} in
  sTest.Tr1 (w 駅 1, w 駅 2)
  ;
end
UseFare0
```

5　参考文献、索引

VDM++[2] は、1970 年代中頃に IBM ウィーン研究所で開発された VDM-SL[1] を拡張し、さらにオブジェクト指向拡張した形式仕様記述言語である。

参考文献

[1] Kyushu University. VDM-SL 言語マニュアル. Kyushu University, 第 2.0 版, 2016. Revised for VDMTools V9.0.2.

[2] Kyushu University. VDM++ 言語マニュアル. Kyushu University, 第 2.0 版, 2016. Revised for VDMTools V9.0.2.

索引