

FHashtable ライブラリー

佐原伸日本フィット株式会社
情報技術研究所

TEL : 03-3623-4683

shin.sahara@jfits.co.jp

平成 16 年 3 月 5 日

概 要

ハッシュ表に関わる関数を提供するモジュールである。

0.1 FHashtable

ハッシュ表に関わる関数を定義する。

class *FHashtable*

Put は、aKey と aValue の写をハッシュ表に追加する。

functions

public static

```

1.0 Put[@T1,@T2]:( $\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)$ )  $\rightarrow$  ( $@T1 \rightarrow @T1 \rightarrow$ 
@T1  $\rightarrow$ 
.1  $@T2 \rightarrow$ 
.2 ( $\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)$ )
.3 Put(aHashtable)(aHashCode)(aKey)(aValue)  $\triangleq$ 
.4 let hashCode = aHashCode(aKey) in
.5 if hashCode  $\in$  dom aHashtable
.6 then aHashtable  $\uparrow$  {hashCode  $\mapsto$  (aHashtable(hashCode)  $\uparrow$ 
{aKey  $\mapsto$  aValue})}
.7 else aHashtable  $\sqcup$  {hashCode  $\mapsto$  {aKey  $\mapsto$  aValue}};
```

PutAll は、写像の内容をハッシュ表に追加する。

public static

```

2.0 PutAll[@T1,@T2]:( $\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)$ )  $\rightarrow$  ( $@T1 \rightarrow @T1$ )
 $\rightarrow$ 
.1 ( $@T1 \xrightarrow{m} @T2 \rightarrow$ 
.2 ( $\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)$ )
.3 PutAll(aHashtable)(aHashCode)(aMap)  $\triangleq$ 
.4 PutAllAux[@T1,@T2](aHashtable)(aHashCode)(aMap)(dom aMap);
```

public static

```

3.0 PutAllAux[@T1,@T2]:( $\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)$ )  $\rightarrow$  ( $@T1 \rightarrow \mathbb{Z}$ )
 $\rightarrow$  ( $@T1 \xrightarrow{m} @T2$ )  $\rightarrow$ 
.1  $@T1\text{-set} \rightarrow$ 
.2 ( $\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)$ )
.3 PutAllAux(aHashtable)(aHashCode)(aMap)(aKeySet)  $\triangleq$ 
.4 if aKeySet = {}
.5 then aHashtable
.6 else let aKey  $\in$  aKeySet in
.7 let  $\check{5}B0Hashtable = \text{Put}[@T1,@T2](aHashtable)(aHashCode)(aKey)(aMap(aKey))$  in
.8 PutAllAux[@T1,@T2]( $\check{5}B0Hashtable$ )(aHashCode)(aMap)(aKeySet \
{aKey});
```

Get は、aKey に対応する値を取り出す。

public static

```

4.0  $Get[@T1, @T2] : (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)) \rightarrow (@T1 \rightarrow @T1) \rightarrow @T1 \rightarrow [@T2]$ 
.1  $Get(aHashtable)(aHashCode)(aKey) \triangleq$ 
.2   let  $hashcode = aHashCode(aKey)$  in
.3   if  $hashcode \in \text{dom } aHashtable$ 
.4   then  $FMap' Get[@T1, @T2](aHashtable(hashcode))(aKey)$ 
.5   else nil ;

```

Remove は、key とそれに対応する値をハッシュ表から削除する。

public static

```

5.0  $Remove[@T1, @T2] : (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)) \rightarrow (@T1 \rightarrow @T1) \rightarrow @T1 \rightarrow (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2))$ 
.1  $Remove(aHashtable)(aHashCode)(aKey) \triangleq$ 
.2   let  $hashcode = aHashCode(aKey)$  in
.3    $\{h \mapsto (\{aKey\} \triangleleft aHashtable(hashcode)) \mid h \in \{hashcode\}\} \sqcup$ 
.4    $\{hashcode\} \triangleleft aHashtable;$ 

```

Clear は、ハッシュ表をクリアする。

public static

```

6.0  $Clear[@T1, @T2] : () \rightarrow (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2))$ 
.1  $Clear() \triangleq$ 
.2    $(\{\mapsto\});$ 

```

KeySet は、ハッシュ表のすべての key の集合を返す。

public static

```

7.0  $KeySet[@T1, @T2] : (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)) \rightarrow @T1\text{-set}$ 
.1  $KeySet(aHashtable) \triangleq$ 
.2   let  $aMapSet = \text{rng } aHashtable,$ 
.3    $f = \lambda x : @T1 \xrightarrow{m} @T2 \cdot \text{dom } x$  in
.4   if  $aMapSet \neq \{\}$ 
.5   then  $\bigcup FSet' Fmap[@T1, @T2](f)(aMapSet)$ 
.6   else  $\{\}$ ;

```

ValueSet は、Hashtable のすべての値の集合を返す。

public static

```

8.0  $ValueSet[@T1, @T2] : (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)) \rightarrow @T2\text{-set}$ 
.1  $ValueSet(aHashtable) \triangleq$ 
.2   let  $aMapSet = \text{rng } aHashtable,$ 
.3    $f = \lambda x : @T1 \xrightarrow{m} @T2 \cdot \text{rng } x$  in
.4   if  $aMapSet \neq \{\}$ 
.5   then  $\bigcup FSet' Fmap[@T1, @T2](f)(aMapSet)$ 
.6   else  $\{\}$ ;

```

Size は、Hashtable 中の key の数を返す。

public static

9.0 $Size[@T1, @T2] : (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)) \rightarrow \mathbb{N}$

- .1 $Size(aHashtable) \triangleq$
- .2 $card\ KeySet[@T1, @T2](aHashtable);$

IsEmpty は、Hashtable 中に key が無いかなを返す。

public static

10.0 $IsEmpty[@T1, @T2] : (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)) \rightarrow \mathbb{B}$

- .1 $IsEmpty(aHashtable) \triangleq$
- .2 $KeySet[@T1, @T2](aHashtable) = \{\};$

Contains は、与えられた aValue があるならば、true を返す。

public static

11.0 $Contains[@T1, @T2] : (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)) \rightarrow @T2 \rightarrow \mathbb{B}$

- .1 $Contains(aHashtable)(aValue) \triangleq$
- .2 $let\ aMapSet = rng\ aHashtable\ in$
- .3 $if\ aMapSet \neq \{\}$
- .4 $then\ \exists\ aMap \in aMapSet \cdot aValue \in rng\ aMap$
- .5 $else\ false;$

ContainsKey は、与えられた key があるならば、true を返す。

public static

12.0 $ContainsKey[@T1, @T2] : (\mathbb{Z} \xrightarrow{m} (@T1 \xrightarrow{m} @T2)) \rightarrow @T1 \rightarrow \mathbb{B}$

- .1 $ContainsKey(aHashtable)(aKey) \triangleq$
- .2 $let\ aMapSet = rng\ aHashtable\ in$
- .3 $if\ aMapSet \neq \{\}$
- .4 $then\ \exists\ aMap \in aMapSet \cdot aKey \in dom\ aMap$
- .5 $else\ false$

end FHashtable

Test Suite : vdm.tc

Class : FHashtable

Name	#Calls	Coverage
FHashtable'Get	5	94%
FHashtable'Put	21	✓
FHashtable'Size	4	80%
FHashtable'Clear	1	✓
FHashtable'KeySet	10	88%
FHashtable'PutAll	4	90%
FHashtable'Remove	5	✓
FHashtable'IsEmpty	2	83%
FHashtable'Contains	7	92%
FHashtable'ValueSet	4	88%

Name	#Calls	Coverage
FHashtable'PutAllAux	17	93%
FHashtable'ContainsKey	3	92%
Total Coverage		93%

0.2 FHashtableT

FHashtable のテストを行う。

class *FHashtableT*

functions

public static

```
13.0  run : () → ℬ*
      .1  run ()  $\triangleq$ 
      .2    let testcases = [t1, t2, t3, t4, t5, t6] in
      .3      FSequence'Fmap[FTestDriver'TestCase*, ℬ] (FTestDriver'run) (testcases)
```

0.2.1 Contains, PutAll を検査する

values

```
14.0  t1 = [
      .1      mk-FTestDriver'TestCase
      .2      (
      .3        "FHashtableT01 : \tContains, PutAll",
      .4        let aHashCode = λ x : ℤ · x mod 13,
      .5        p1 = FHashtable'PutAll[ℤ, char*] ({↦}) (aHashCode)
      .6        (
      .7          {1 ↦ "Sahara", 2 ↦ "Sato", 14 ↦
" Sakoh"}),
      .8          c1 = FHashtable'Contains[ℤ, char*] (p1) in
      .9          c1 ("Sahara") ∧
      .10         c1 ("Sato") ∧
      .11         c1 ("Sakoh") ∧
      .12         c1 ("") = false)];
```

0.2.2 Clear, Remove, ContainsKey を検査する

```
15.0  t2 = [  
.1      mk-FTestDriver' TestCase  
.2      (  
.3          "FHashtableT02 : \tClear, Remove, ContainsKey",  
.4          let aHashCode = λ x : char* · if x = ""  
.5              then ""  
.6              else FSequence' Take[char] (1) (x),  
.7          h2 = FHashtable' PutAll[char*, ℤ] ({↦}) (aHashCode)  
.8              (  
.9                  {"a" ↦ 1, "b" ↦ 2, "c" ↦ 3}),  
10.0      h3 = FHashtable' Clear[ℤ, char*] (),  
11.0      deletedh2 = FHashtable' Remove[char*, ℤ] (h2) (aHashCode) ("b"),  
12.0      c1 = FHashtable' Contains[char*, ℤ] (deletedh2),  
13.0      ck1 = FHashtable' ContainsKey[char*, ℤ] (deletedh2) in  
14.0      h3 = {↦} ∧  
15.0      FHashtable' Get[char*, ℤ] (deletedh2) (aHashCode) ("b") =  
nil ∧  
16.0      c1 (2) = false ∧  
17.0      c1 (1) ∧  
18.0      c1 (3) ∧  
19.0      ck1 ("b") = false ∧  
20.0      ck1 ("a") ∧  
21.0      ck1 ("c")]);
```

0.2.3 Put, Get を検査する

```
16.0  t3 = [  
.1      mk-FTestDriver' TestCase  
.2      (  
.3          "FHashtableT03 : \tPut, Get",  
.4          let aHashCode = λ x : ℤ · x mod 13,  
.5          put = FHashtable' Put[ℤ, char*],  
.6          p1 = put ({↦}) (aHashCode) (1) ("Sahara"),  
.7          p2 = put (p1) (aHashCode) (2) ("Bush"),  
.8          p3 = put (p2) (aHashCode) (2) ("Sato"),  
.9          p4 = put (p3) (aHashCode) (14) ("Sakoh"),  
10.0      get = FHashtable' Get[ℤ, char*] (p4) in  
11.0      get (aHashCode) (1) = "Sahara" ∧  
12.0      get (aHashCode) (2) = "Sato" ∧  
13.0      get (aHashCode) (14) = "Sakoh" ∧  
14.0      get (aHashCode) (99) = nil ];
```

0.2.4 KeySet, ValueSet を検査する

```
17.0 t4 = [  
.1     mk-FTestDriver' TestCase  
.2     (  
.3         "FHashtableT04 : \tKeySet, ValueSet",  
.4         let aHashCode =  $\lambda x : \mathbb{Z} \cdot x \bmod 13$ ,  
.5         put = FHashtable' Put[ $\mathbb{Z}$ , char*],  
.6         p1 = put ({ $\mapsto$ }) (aHashCode) (1) ("Sahara"),  
.7         p2 = put (p1) (aHashCode) (2) ("Bush"),  
.8         p3 = put (p2) (aHashCode) (2) ("Sato"),  
.9         p4 = put (p3) (aHashCode) (14) ("Sakoh"),  
.10        k = FHashtable' KeySet[ $\mathbb{Z}$ , char*],  
.11        v = FHashtable' ValueSet[ $\mathbb{Z}$ , char*] in  
.12        k (p1) = {1}  $\wedge$   
.13        v (p1) = {"Sahara"}  $\wedge$   
.14        k (p2) = {1, 2}  $\wedge$   
.15        v (p2) = {"Sahara", "Bush"}  $\wedge$   
.16        k (p4) = {1, 2, 14}  $\wedge$   
.17        v (p4) = {"Sahara", "Sato", "Sakoh"}]);
```

0.2.5 hashCode が重複する場合を検査する

```
18.0 t5 = [  
.1     mk-FTestDriver' TestCase  
.2     (  
.3         "FHashtableT05 : \tSameHashCode",  
.4         let aHashCode1 =  $\lambda x : \mathbb{Z} \cdot x \bmod 13$ ,  
.5         h1 = FHashtable' PutAll[ $\mathbb{Z}$ , char*] ({ $\mapsto$ }) (aHashCode1)  
.6         (  
.7             {1  $\mapsto$  "SaharaShin", 2  $\mapsto$   
"SatoKei", 14  $\mapsto$  "SakohHiroshi", 27  $\mapsto$  "NishikawaNoriko"}),  
.8             h2 = FHashtable' Remove[ $\mathbb{Z}$ , char*] (h1) (aHashCode1) (14) in  
.9             FHashtable' KeySet[ $\mathbb{Z}$ , char*] (h2) = {1, 2, 27}  $\wedge$   
.10            FHashtable' ValueSet[ $\mathbb{Z}$ , char*] (h2) =  
{ "SaharaShin", "SatoKei", "NishikawaNoriko" });
```


0.2.6 Size を検査する

```
19.0  t6 = [  
.1      mk-FTestDriver 'TestCase  
.2      (  
.3          "FHashtableT06 : \tSize",  
.4          let aHashCode1 =  $\lambda x : \mathbb{Z} \cdot x \bmod 13$ ,  
.5              remove = FHashtable 'Remove [ $\mathbb{Z}$ , char*],  
.6              h1 = FHashtable 'PutAll [ $\mathbb{Z}$ , char*] ( $\{\mapsto\}$ ) (aHashCode1)  
.7                  (  
.8                      {1       $\mapsto$       "SaharaShin", 2       $\mapsto$   
"SatoKei", 14  $\mapsto$  "SakohHiroshi" }},  
.9                      h2 = remove (h1) (aHashCode1) (1),  
.10                     h3 = remove (h2) (aHashCode1) (2),  
.11                     h4 = remove (h3) (aHashCode1) (14),  
.12                     isempty = FHashtable 'IsEmpty [ $\mathbb{Z}$ , char*],  
.13                     size = FHashtable 'Size [ $\mathbb{Z}$ , char*] in  
.14                     isempty (h4)  $\wedge$   
.15                     size (h4) = 0  $\wedge$   
.16                     isempty (h3) = false  $\wedge$   
.17                     size (h3) = 1  $\wedge$   
.18                     size (h2) = 2  $\wedge$   
.19                     size (h1) = 3)]  
  
end FHashtableT
```