Forritunarmál Hópverkefni 9

Ragnar Björn Ingvarsson, rbi3
Daníel Snær Halldórsson, dsh11
Ólafur Sær Sigursteinsson, oss27

24. október 2024

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;; Design document
;;; ===============
;;;
;;; Exported
;;; --------
;;;
;;; Use:   val s = makeSet();
;;; Pre:   Nothing.
;;; Post:  s contains a new empty set of
;;;        values that are allowed as
;;;        arguments to the imported
;;;        function comp.
;;;
;;; Imported
;;; --------
;;;
;;; Use:  val c = comp(x,y);
;;; Pre:  x and y are values that are
;;;       allowed to be stored in the sets
;;;       implemented here.
;;; Post: c is an integer that is <0 if x
;;;       must precede y, >0 if y must
;;;       precede x, and ==0 if x and y
;;;       are equal.
;;; Note: comp should define an ordering on
;;;       the values allowed in the sets.
;;;       The ordering should ensure that
;;;       any finite set of values has a
;;;       least element.
;;;
;;; Use:  s.add(x);
;;; Pre:  s is a set that can contain x.
;;; Post: x has been added to s if it was
;;;       not already in s. If x was
;;;       was already in s then s is
;;;       unchanged.
;;;
;;; Use:  val e = s.isEmpty();
;;; Pre:  s is a set.
;;; Post: e contains true if s is empty,
;;;       false otherwise.
;;;
;;; Use:  val c = s.contains(x);
```

```
;;; Pre:  s is a set that can contain x.
;;; Post: c is true if s contains x, false
;;;       otherwise.
;;;
;;; Use:  val m = s.min();
;;; Pre:  s is a set, not empty.
;;; Post: m is the minimal value in s,
;;;       according to the imported
;;;       function comp.
;;;
;;; Use:  s.remove(x);
;;; Pre:  s is a set that can contain x.
;;; Post: If s contained x then x has
;;;       been removed from s, otherwise
;;;       s is unchanged.
;;;
;;; Use:  val r = s.mapReduce(op,f,u);
;;; Pre:  s is a set.
;;;       op is a binary function,
;;;       f is a unary function.
;;;       u is some value such that
;;;       the expression in the post-
;;;       condition can be computed.
;;; Post: The expression
;;;        u ! f(x1) ! f(x2) ! ... ! f(xN)
;;;       has been computed, where x!y
;;;       is equivalent to op(x,y) and
;;;       the computation is performed
;;;       from left to right, and the
;;;       values x1,x2,...,xN are all the
;;;       values in s in ascending order.
;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

"set.mmod" =
{{
makeSet = fun makeSet();
}}
*
!
{{
makeSet =
    obj()
    {
```

```
var set = [];

;;; Data invariant:
;;;    A list of values that are allowed as arguments
;;;    to the comp function in ascending order according
;;;    to the ordering of comp.

msg add(x)
{
    ;;; Use:  z = help(x,l)
    ;;; Pre:  x is a value allowed as an argument to
    ;;;       the imported function comp.
    ;;;       l is a list of values allowed as
    ;;;       arguments to the imported function comp
    ;;;       that can contain x.
    ;;; Post: z contains the list l where x has been added
    ;;;       to the list if it did not contain x beforehand.
    ;;;       Otherwise z contains l unchanged.
    rec fun help(x, l)
    {
        if ( l == [] || comp(head(l),x) > 0 )
        {
            return x:l;
        }
        elsif ( head(l) == x )
        {
            return l;
        };
        head(l) : help(x, tail(l));
    };
    set = help(x, set);
};

msg isEmpty()
{
    if ( set == [] ) { return true };
    return false;
};

msg contains(x)
{
    ;;; Use:  z = help(x,l)
    ;;; Pre:  l is a list that can contain the value x
    ;;; Post: z contains true if l contains x, false otherwise.
    rec fun help(x, l)
```

```
    {
        if ( l == [] ) { return false; }
        elsif ( head(l) == x ) { return true; };
        return help(x, tail(l));
    };
    return help(x, set);
};

msg min()
{
    ;;; Use:  z = help(l,min)
    ;;; Pre:  l is a list of values allowed as
    ;;;       arguments to the imported function comp.
    ;;;       min is a value allowed as an argument to comp
    ;;;       and that l can contain.
    ;;; Post: z is the minimum value according to comp of
    ;;;       all the values of both l and min.
    rec fun help(l, min)
    {
        if ( l == [] )
        {
            return min;
        }
        elsif ( comp(head(l), min) < 0 )
        {
            return help(tail(l), head(l));
        };
        return help(tail(l), min);
    };
    return help(tail(set), head(set));
};

msg remove(x)
{
    ;;; Use:  z = help(x,l)
    ;;; Pre:  l is a list that can contain x
    ;;; Post: If l contains x then z contains l where
    ;;;       x has been removed, otherwise z contains l.
    rec fun help(x, l)
    {
        if ( l == [] )
        {
            return l;
        }
        elsif ( head(l) == x )
```

```
            {
                return tail(l);
            };
            return head(l) : help(x, tail(l));
        };
        set = help(x, set);
    };

    msg mapReduce(op,f,u)
    {
        ;;; Use:  z = help(x,op,f,u)
        ;;; Pre:  x = [x1,x2,...,xN] is a list of values
        ;;;       allowed as arguments to f.
        ;;;       op is a binary function, f is a unary
        ;;;       function and u is a value such that the value
        ;;;       in the post-condition can be evaluated.
        ;;; Post: z contains the value of the expression
        ;;;       u ! f(x1) ! f(x2) ! ... ! f(xN)
        ;;;       where x!y is equivalent to op(x,y) and
        ;;;       the computation is performed
        ;;;       from left to right.
        rec fun help(x, op, f, u)
        {
            if ( x == [] )
            {
                return u;
            };
            return help( tail(x), op, f, op(u, f(head(x))));
        };
        return help(set, op, f, u);
    };
};
}}
;
```

```
ragnar@gamer ~/school/forritun/v9 ◆ java -jar morpho.jar testset
false
45
123456789
true
ragnar@gamer ~/school/forritun/v9 ◆ java -jar morpho.jar testset2
false
45
[1][2][3][4][5][6][7][8][9]
true
ragnar@gamer ~/school/forritun/v9 ◆
```