```
library libraryBag

from Basic/Numbers get Int

spec Bag0[sort Elem] given Int =
        generated type Bag[Elem] ::= bagVide | ajouter(Bag[Elem]; Elem)
        op      frequence : Bag[Elem] * Elem -> Int
        forall x, y : Elem; M, N : Bag[Elem]
        . frequence(bagVide, y) = 0
        . frequence(ajouter(M, x), y)
           = 1 + frequence(M, y) when x = y else frequence(M, y)
        . M = N
           <=> forall x : Elem . frequence(M, x) = frequence(N, x)
end

spec Bag[sort Elem] given Int =
        Bag0[sort Elem]
then preds estVide : Bag[Elem];
                appartient : Elem * Bag[Elem];
                inclut : Bag[Elem] * Bag[Elem]
        op      enlever : Bag[Elem] * Elem -> Bag[Elem]
        forall x, y : Elem; M, N : Bag[Elem]
        . estVide(M) <=> M = bagVide
        . appartient(x, M) <=> frequence(M, x) > 0
        . inclut(M, N)
           <=> forall x : Elem . frequence(M, x) <= frequence(N, x)
        . enlever(M, x) = N
           <=> forall y : Elem
                        . (frequence(N, y) = frequence(M, x) - 1 if x = y)
                          /\ (frequence(N, y) = frequence(M, y) if not x = y)
end
```