

# Tutorial 4: Bag

Ken Robinson

26th May 2000

## Contents

<b>1</b>	<b>A Bag machine</b>	<b>2</b>
1.1	Using Abstract Constants . . . . .	2
1.2	Animating the Bag machine . . . . .	2
1.3	The Specification . . . . .	3

# 1 A Bag machine

A *bag* is a mathematical aggregate construct that can contain multiple instances of a value, but for which there is no ordering. As an example of a bag consider a bag containing jelly beans: the bag will contain a certain number of red jelly beans, black jelly beans etc. Generally with bags we do not explicitly represent zero occurrences of an item: the bag either contains some black jelly beans, or it doesn't.

Develop a machine *Bag* with parameters *VALUE* and *maxbag* being the set of values to be included in bags and the maximum number of bags, respectively.

Model bags as partial functions in  $VALUE \rightarrow \mathbb{N}_1$ .

Provide the following operations:

1.  $bag \leftarrow NewBag$ : create a new bag.
2.  $DeleteBag(bag)$ : delete an existing bag.
3.  $AddItem(bag, item)$ : add *item* (in set *VALUE*) to the bag, *bag*.
4.  $in \leftarrow inBag(item, bag)$ : return *TRUE* or *FALSE* depending on whether *item* is in *bag* or not, respectively.
5.  $count \leftarrow Count(item, bag)$  return the number of occurrences of *item* in *bag*, zero if it is not in the bag.
6.  $BagUnion(bag1, bag2)$ : form the bag union of *bag1* and *bag2* with the result in *bag1*. Note: in bag union the frequencies should be summed.

## 1.1 Using Abstract Constants

This **Bag** machine uses abstract constant functions, *emptyBag* and *bagCount* defined as follows:

$$\begin{aligned} emptyBag &\in VALUE \rightarrow \{ 0 \} \wedge \\ bagCount &\in Bag ( VALUE ) \rightarrow ( VALUE \rightarrow \mathbb{N} ) \wedge \\ \forall bb . ( bb \in Bag ( VALUE ) \Rightarrow bagCount ( bb ) = emptyBag \triangleleft bb ) \end{aligned}$$

where  $Bag(X) \hat{=} X \rightarrow \mathbb{N}_1$

The function *bagCount* converts the partial function model of bag to a total function. This gets around the problem of having to check the domain of the bag before applying the bag to a value in order to obtain the frequency of the value in the bag. Given a bag *b* and a value *v*,  $bagCount(b)(v)$  will yield the frequency of *v* in the bag. If *v* is not in  $dom(b)$  then the frequency is given by  $emptyBag(v)$ , which is 0 for all *v*.

The function *bagCount* considerably simplifies the specification of the operation **BagUnion**, but it is also used to simplify **AddItem**.

## 1.2 Animating the Bag machine

A theory file, `Bag.mch.thy`, has been defined that assists the animator interpreting the constant functions, which are left as *deferred*.

### 1.3 The Specification

**MACHINE** *Bag* ( *VALUE* , *maxbag* )

**CONSTRAINTS** *maxbag*  $\in \mathbb{N}_1$

**SEES** *Bool\_TYPE*

**SETS** *BAG*

**CONSTANTS**

*emptyBag* ,  
*bagCount*

**PROPERTIES**

*emptyBag*  $\in VALUE \rightarrow \{ 0 \} \wedge$   
*bagCount*  $\in Bag ( VALUE ) \rightarrow ( VALUE \rightarrow \mathbb{N} ) \wedge$   
 $\forall bb . ( bb \in Bag ( VALUE ) \Rightarrow bagCount ( bb ) = emptyBag \triangleleft bb )$

**VARIABLES**

*bags* ,  
*freq*

**INVARIANT**

*bags*  $\subseteq BAG \wedge$   
*freq*  $\in bags \rightarrow Bag ( VALUE )$

**INITIALISATION**

*bags* , *freq* := { } , { }

**OPERATIONS**

*bag*  $\leftarrow$  **NewBag**  $\hat{=}$   
**PRE** *bags*  $\neq BAG$   
**THEN ANY** *bb*  
**WHERE** *bb*  $\in BAG - bags$   
**THEN** *bags* := *bags*  $\cup$  { *bb* } || *freq* ( *bb* ) := { } || *bag* := *bb*  
**END**  
**END ;**

**DeleteBag** ( *bag* )  $\hat{=}$   
**PRE** *bag*  $\in bags$   
**THEN** *bags* := *bags* - { *bag* } || *freq* := { *bag* }  $\triangleleft$  *freq*  
**END ;**

**AddItem** ( *bag* , *item* )  $\hat{=}$   
**PRE** *bag*  $\in bags \wedge item \in VALUE$   
**THEN** *freq* ( *bag* ) ( *item* ) := *bagCount* ( *freq* ( *bag* ) ) ( *item* ) + 1  
**END ;**

```

in ← inBag ( bag , item ) ≐
  PRE  bag ∈ bags ∧ item ∈ VALUE
  THEN in := bool ( item ∈ dom ( freq ( bag ) ) )
  END  ;

count ← Count ( bag , item ) ≐
  PRE  bag ∈ bags ∧ item ∈ VALUE
  THEN count := bagCount ( freq ( bag ) ) ( item )
  END  ;

bagUnion ( bag1 , bag2 ) ≐
  PRE  bag1 ∈ bags ∧ bag2 ∈ bags
  THEN ANY bag3
    WHERE bag3 ∈ Bag ( VALUE ) ∧
      ∀ item . ( item ∈ VALUE ⇒
        bagCount ( bag3 ) ( item ) =
          bagCount ( freq ( bag1 ) ) ( item ) +
          bagCount ( freq ( bag2 ) ) ( item ) )
    THEN freq ( bag1 ) := bag3
    END
  END

```

#### DEFINITIONS

$Bag ( X ) \hat{=} X \leftrightarrow \mathbb{N}_1$

END