BAGS

This page is intentionally blank

☺

INTRODUCTION

- A *bag* (also called a *family* or *multi-set*) is an extension to the idea of a *set*
    - in a *bag*, multiple occurrences of elements are allowed

- Like a set, a bag may be described by enumeration; but the enclosing brackets are now "outlined" square brackets:

  ⟦ Tom, Mary, Manesh, Tom, Henry, Mary ⟧

    - An empty bag is shown as ⟦ ⟧

- As with sets, the order of enumeration in bags is immaterial; but, for two bags to be equal, they each must contain the same set of elements and the number of occurrences of each element must be the same

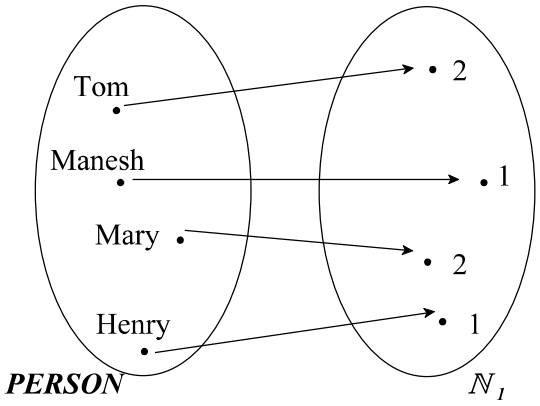
BAGS AS FUNCTIONS

- The bag shown on the previous page:

  $$[\![\text{Tom, Mary, Manesh, Tom, Henry, Mary}]\!]$$

  is equivalent to the mapping:

  $$\{ \text{Tom} \mapsto 2, \text{Mary} \mapsto 2, \text{Henry} \mapsto 1, \text{Manesh} \mapsto 1 \}$$



- In general, given some set X:

  $$bag\ X == X \nrightarrow \mathbb{N}_1$$

  - which says a bag with elements from a set X can be modelled as a partial function from the type of the elements of X to $\mathbb{N}_1$

BAG OPERATORS

- *count* determines the number of times a specified element occurs in a particular bag

- A generic definition of *count* might be:

$$[X]$$

$$count : \text{bag } X \rightarrowtail\!\!\!\!\!\twoheadrightarrow (X \rightarrow \mathbb{N})$$

$$\forall B : \text{bag } X \bullet count\ B = \{ x : X \bullet x \mapsto 0 \} \oplus B$$

  where $\rightarrowtail\!\!\!\!\!\twoheadrightarrow$ represents a *bijection*

- If the *bag* used previously as an example, represents people winning raffle prizes:

  Raffle_Winners = =
  ⟦Tom, Mary, Manesh, Tom, Henry, Mary⟧

  then, using function application, the *count* function reveals how many prizes each winner has gained:

  count Raffle_Winners Tom = 2
  count Raffle_Winners Henry = 1
  count Raffle_Winners Anne = 0

BAG OPERATORS

- For bags, *in* behaves like $\in$ does for sets
  - e.g. Mary *in Raffle_Winners* = True
    James *in Raffle_Winners* = False
    $\neg$(James *in Raffle_Winners*) = True

- *Bag union* (also called *bag-sum*) is, similarly, analogous to set union (but with a difference!)
  - The symbol for bag union is $\uplus$ and its effects are illustrated by:

    Raffle_Winners $\uplus$ $[\![$Tom, Mary, Jane$]\!]$ =
    $\{$ Tom $\mapsto$ 3, Mary $\mapsto$ 3, Henry $\mapsto$ 1,
    Manesh $\mapsto$ 1, Jane $\mapsto$ 1 $\}$

- It is important to realize that
  - if X is a set and $x \in$ X, then X $\cup$ $\{x\}$ = X, but
  - if B is a bag and *x in* B, then B $\uplus$ $[\![x]\!]$ $\neq$ B

BAG OPERATORS

- The bag function *items*, creates a bag from a sequence

- If S is the sequence $\langle$Fred, Dick, Jo, Fred, Jo, Jo$\rangle$ then

$$items\ S = = \{ Fred \mapsto 2, Dick \mapsto 1, Jo \mapsto 3 \}$$
$$= = [\![Fred, Fred, Dick, Jo, Jo, Jo]\!]$$

- While further bag operators are also available those described here are regarded as the "standard" bag operators and are sufficient for most purposes

- Other operators (e.g. bag subtraction, bag multiplication, etc) are described in [Hayes, 1989] and [Spivey, 1992]

SUMMARY OF SYMBOLS

$[\![a, b, .. , n]\!]$	*the bag containing elements* a, b, .. , n
bag X	*the bag of elements which occur in the set* X ($= = X \pfun \mathbb{N}_1$)
count B *x*	*the number of times the element x occurs in the bag* B
x **in** B	*True if x is an element of bag* B*; otherwise False*
$B_1 \uplus B_2$	*The union of bags* B_1 *and* B_2

REFERENCES

[Hayes, 1989]
Hayes I, "A Generalisation of Bags in Z", *Z User Workshop*, Oxford, 4(6): 113-127, Springer Verlag, 1989

[Spivey, 1992]
Spivey J M, *The Z Notation: A Reference Manual*, 2nd edn. Hemel Hempstead: Prentice Hall, 1992

EXERCISES

1. Express the following using bag notation:

   (a) $\{ \text{beans} \mapsto 3, \text{pasta} \mapsto 5, \text{cheese} \mapsto 2 \}$
   (b) $\{ x : \mathbb{N} \mid x \leq 4 \bullet x \mapsto x+1 \}$

---

2. Express the following bags using maplet notation:

   (a) ⟦ fig, pear, plum, fig, plum, pear, pear, fig ⟧
   (b) ⟦ 4, 5, 3, 2, 4, 3, 7, 6, 4, 3, 2, 1 ⟧

---

3. Express the following bags using set comprehension:

   (a) ⟦ 1, 1, 2, 2, 2, 3, 3, 3, 3 ⟧
   (b) ⟦ 3, 3, 0, 4, 2, 2, 1, 1, 2, 1, 1, 0, 0, 0, 0 ⟧

---

4. A computer network offers users a variety of services. Unfortunately the amount of available storage is almost exhausted and since funds are not available for upgrades or enhancements, the network administrator wishes to remove services which are rarely or never used. To achieve this, the administrator needs to ascertain exactly how many times each service is accessed.

   If each service is identified by a unique integer number (i.e. *service* : $\mathbb{N}$) and, each time a particular service is accessed, its identifier is logged, show how *bags* could be used to specify what is required.

---

5. The *size* of a bag is the total number of elements it contains irrespective of repetitions. A couple of examples should clarify the concept:

   size ⟦ ⟧ = 0, and
   size ⟦apple, fig, pomegranate, fig, pear, fig⟧ = 6

   It seems obvious that the size of any bag is given by the sum of the numbers in the range of the bag and, since "+" can only be used as a binary operator for addition, it will have to be applied successively to pairs of numbers taken from the range of the bag. This suggests a recursive definition can be created with *size* ⟦ ⟧ = 0 providing the "base" case.

   Create such a generic recursive definition for the function *size* where *size* : bag X → $\mathbb{N}$

6. Assuming the existence of the *count* function create a generic definition for the function *items* where *items* : **seq** X → bag X.

---

7. If x : X and b : bag X, the infix *in* bag "operator" can be regarded as a relation over X and bag X (i.e. $\_ \ in \_ : X \leftrightarrow$ bag X). Provide a formal generic definition of the *in* bag operator.

---

8. Sorting may be viewed as an activity that accepts a sequence of items of some type as input and provides a sequence of the same items as output. The output sequence must, of course, have the assured quality of "order" and we might choose to define the sorting process by:

$$Sort\_ascending == \{ in?, out! : \mathbf{seq}\ X \mid \forall i, j : \mathbf{dom}\ out! \bullet i < j \Rightarrow (out!\ i \leq out!\ j) \}$$
$$Sort\_descending == \{ in?, out! : \mathbf{seq}\ X \mid \forall i, j : \mathbf{dom}\ out! \bullet i < j \Rightarrow (out!\ i \geq out!\ j) \}$$

Although these predicates capture the essential nature of sorting they do not provide a complete description. For example, the given predicates do not describe the fact that the sequence *out*! must contain the same elements, in the same frequency, as the sequence *in*?.

Amend the given predicates to take account of this "conservation of elements" property.
