Department of Computer Science, CUI Lahore Campus

Formal Methods

By

Farooq Ahmad

Bags in Z

Concept of bags, bag data type, operators for bags

Topics covered

- Bags
- Bag data type
- Declaring bag variable
- Operators for bag variables

Concept of a bag

- We write [[a_1, … , a_n]] for the bag { (a_1, k_1), … ,(a_n, K_n)}, where for each i the element ai appears ki times in the list a1, .. , an.
- *bag* X is the set of bags or multisets of elements of X. These are collections of elements of X in which the number of times an element occurs is significant.
- The empty bag [[ ]] is a notation for the empty function from X to N
- The number of times x appears in the bag B is *count* B x.

Bags

- If we wish to record multiplicities, but not ordering, then we may use bag.
- We write ⟦ a, a, b, b, c, c ⟧ to denote the bag containing two copies of a, two copies of b, and two copies of c.
- The order in which elements are written is not important
- The expression
  ⟦ a, a, b, b, c, c ⟧ equals ⟦ a, b, b, a, c, c ⟧

Bags as partial function

- If B is a bag of elements from set X, then B may be regarded as a partial function from X to N.
- Any element of X in B is associated with natural number, recording number of instances in it.

Example: $[\![a, a, b, b, c, c]\!] = \{a \mapsto 2, b \mapsto 2, c \mapsto 2\}$, each element associated with the number 2.

- If X is a set, then set of all bags from X may be denoted by the following generic abbreviation:

$$\text{bag } X == X \rightarrow\!\!\!\!\!\!+ \; \mathbb{N} \setminus \{0\}$$

Declaring bag variable

- Let Product be a set of products sold on a store, where [Product] is a type
- bag Product == Product $\nrightarrow$ $\mathbb{N}$
- Bag Product is a set of all bags of products.
- Now, stock : bag Product is a bag variable representing stock in a store.
- Stock = {(glass, 100), (cup, 200), (plate, 200)}

Count of a bag

The number of times x appears in the bag B is
***count* B x** OR **B#x**

count B x ==

count ⟦ a, a, b, b, b, c, c, b, a ⟧ a = 3
count ⟦ a, a, b, b, b, c, c, b, a ⟧ b = 4
count ⟦ a, a, b, b, b, c, c, b, a ⟧ c = 2

OR

B # x ==

⟦ a, a, b, b, b, c, c, b, a ⟧ # a = 3
⟦ a, a, b, b, b, c, c, b, a ⟧ # b = 4
⟦ a, a, b, b, b, c, c, b, a ⟧ # c = 2

Operators for bag

$\uplus$	–	Bag union
$\uminus$	–	Bag difference

$B \uplus C$ is the *bag union* of B and C: the number of times any object appears in $B \uplus C$ is the sum of the number of times it appears in B and in C. $B \uminus C$ is the *bag difference* of B and C: the number of times any object appears in it is the number of times it appears in B minus the number of times it appears in C, or zero if that would be negative.

Example of bag union operator

- {(glass, 100), (cup, 200), (plate, 200)} ⊎ {(glass, 50), (cup, 100), (spoon, 500)}
- = {(glass, 150), (cup, 300), (plate, 200), (spoon, 500)}

Bag difference operator

- The required product quantities are removed from the available stock using the bag difference operator ($\uminus$ the set difference operator '\' for sets). Here, for example, $\{nuts \longmapsto 5, bolts \longmapsto 6\} \uminus \{nuts \longmapsto 3\}$ would result in
- $\{nuts \longmapsto 2, bolts \longmapsto 6\}$.

Sub Bag

- $\sqsubseteq$ is the sub-bag relational operator from the Z tool kit. This ensures a precondition that there are enough quantities of the required product(s) in stock.
- For example, $\{nuts \mapsto 3\} \sqsubseteq \{nuts \mapsto 5, bolts \mapsto 6\}$ is true.

Summary of the lecture: Conclusion

- Concept of bags in Z
- Bag data type
- Declaring bag variables
- Operators for bags

Reference and reading material

- Chapter 4: Section 4.6 of the book “The Z Notation:
- A Reference Manual
- Second Edition
- J. M. Spivey, Programming Research Group University of Oxford