Agnostic to Petri Nets.

MODULE *Helpers*

EXTENDS *Integers*, *Sequences*

Convert a set to a sequence.

```
SetToSeq(s) ≜ LET
    RECURSIVE Helper(_)
    Helper(s_) ≜
        IF s_ = {}
         THEN ⟨⟩
         ELSE
            LET x ≜ CHOOSE x ∈ s_ : TRUE
            IN   ⟨x⟩ ∘ Helper(s_ \ {x})
IN  Helper(s)
```

Left fold a sequence.

```
FoldLSeq(fn(_, _), initVal, s) ≜ LET
    RECURSIVE Helper(_)
    Helper(s_) ≜
        IF s_ = ⟨⟩
         THEN initVal
         ELSE fn(Head(s_), Helper(Tail(s_)))
IN  Helper(s)
```

Sum a sequence.

```
SumSeq(s) ≜ LET
    add(a, b) ≜ a + b
IN  FoldLSeq(add, 0, s)
```

Sum the counts of a Bag.

```
BagSum(B) ≜ LET
    ks ≜ SetToSeq(DOMAIN B)
    vs ≜ [i ∈ 1 .. Len(ks) ↦ B[ks[i]]]
IN  SumSeq(vs)
```

Bags module's $\oplus$ except it can take invalid "bags" where counts might not be > 0.

```
B1 ⊕ B2  ≜
  [e ∈ (DOMAIN B1) ∪ (DOMAIN B2) ↦
      (IF e ∈ DOMAIN B1 THEN B1[e] ELSE 0)
    + (IF e ∈ DOMAIN B2 THEN B2[e] ELSE 0)]
```