Specifying the operations performed in a Pub

Kevin Blackburn
ICL Enterprise Engineering

March 1995

Abstract

This note was produced a VDM-SL course presented by Peter Gorm Larsen to ICL Enterprise Engineering. The modelling of bags was one of the exercises the attendees (including myself) was confronted with during the course. Because I have a background in the Z specification language I finished this exercise before the other attendees. Thus I was asked to use the definitions of the bags to model the operations performed in a pub with a cellar and a bar. This specification is mainly intended for the purpose of illustrating how bags can be used.

1 Modelling of Bags

module BAG

exports

types Bag;
struct $Elem$

values $baga : Bag$;
$bagb : Bag$

functions $Add : Elem \times Bag \to Bag$;
$Count : Elem \times Bag \to \mathbb{N}$;
$Difference : Bag \times Bag \to Bag$;
$Empty : () \to Bag$;
$In : Elem \times Bag \to \mathbb{B}$;
$Intersection : Bag \times Bag \to Bag$;
$Join : Bag \times Bag \to Bag$;
$Remove : Elem \times Bag \to Bag$;
$SeqToBag : Elem^* \to Bag$;
$Size : Bag \to \mathbb{N}$;
$SubBag : Bag \times Bag \to \mathbb{B}$;
$Union : Bag \times Bag \to Bag$

definitions
types

$Elem = \mathrm{A} \mid \mathrm{B} \mid \mathrm{C} \mid \mathrm{D} \mid \mathrm{E}$;

$Bag = Elem \xrightarrow{m} \mathbb{N}_1$

functions

6.0 $Min : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$

.1 $Min\,(i, j) \triangleq$

.2 if $i < j$

.3 then i

.4 else j;

7.0 $Max : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$

.1 $Max\,(i, j) \triangleq$

.2 if $i > j$

.3 then i

.4 else j;

8.0 $AuxSeqToBag : Elem^* \times Bag \to Bag$

.1 $AuxSeqToBag\,(s, b) \triangleq$

.2 cases s :

.3 $[] \to b$,

.4 $[e] \frown rest \to AuxSeqToBag\,(rest, Add\,(e, b))$

.5 end;

9.0 $Empty : () \to Bag$

.1 $Empty\,() \triangleq$

.2 $\{\mapsto\}$;

10.0 $Add : Elem \times Bag \to Bag$

.1 $Add\,(e, b) \triangleq$

.2 if $e \in \mathsf{dom}\ b$

.3 then $b \dagger \{e \mapsto b\,(e) + 1\}$

.4 else $b \dagger \{e \mapsto 1\}$;

11.0 $Remove : Elem \times Bag \to Bag$

.1 $Remove\,(e, b) \triangleq$

.2 if $e \in \mathsf{dom}\ b$

.3 then if $b\,(e) = 1$

.4 then $\{e\} \lhd b$

.5 else $b \dagger \{e \mapsto b\,(e) - 1\}$

.6 else b;

12.0 $Count : Elem \times Bag \to \mathbb{N}$

.1 $Count\,(e, b) \triangleq$

.2 if $e \in \mathsf{dom}\ b$

.3 then $b\,(e)$

.4 else 0;

```
13.0  In : Elem × Bag → 𝔹
 .1   In (e, b) ≜
 .2     e ∈ dom b;

14.0  Join : Bag × Bag → Bag
 .1   Join (b1, b2) ≜
 .2     {e ↦ Max (Count (e, b1), Count (e, b2)) |
 .3           e ∈ (dom b1 ∪ dom b2)};

15.0  Union : Bag × Bag → Bag
 .1   Union (b1, b2) ≜
 .2     {e ↦ Count (e, b1) + Count (e, b2) |
 .3           e ∈ (dom b1 ∪ dom b2)};

16.0  SubBag : Bag × Bag → 𝔹
 .1   SubBag (b1, b2) ≜
 .2     ∀ e ∈ dom b1 ·
 .3           Count (e, b1) ≤ Count (e, b2);

17.0  Difference : Bag × Bag → Bag
 .1   Difference (b1, b2) ≜
 .2     {e ↦ Count (e, b1) − Count (e, b2) |
 .3           e ∈ dom b1 ·
 .4                Count (e, b1) > Count (e, b2)};

18.0  Size : Bag → ℕ
 .1   Size (b) ≜
 .2     if b = {↦}
 .3     then 0
 .4     else let e ∈ dom b in
 .5          b (e) + Size ({e} ⩤ b);

19.0  Intersection : Bag × Bag → Bag
 .1   Intersection (b1, b2) ≜
 .2     {e ↦ Min (Count (e, b1), Count (e, b2)) |
 .3           e ∈ (dom b1 ∩ dom b2)};

20.0  SeqToBag : Elem* → Bag
 .1   SeqToBag (s) ≜
 .2     AuxSeqToBag (s, Empty ())

values

21.0  baga : Bag = {A ↦ 3, B ↦ 2, C ↦ 4};
```

22.0 $bagb : Bag = \{\mathrm{A} \mapsto 1, \mathrm{C} \mapsto 5, \mathrm{D} \mapsto 4, \mathrm{E} \mapsto 1\}$

end BAG

2 Modelling of a Bar

module BAR

imports

23.0 from BAG

24.0 types Bag;
.1 $Elem$

25.0 values $baga : BAG`Bag$;
.1 $bagb : BAG`Bag$

26.0 functions $Add : BAG`Elem \times BAG`Bag \to BAG`Bag$;
.1 $Count : BAG`Elem \times BAG`Bag \to \mathbb{N}$;
.2 $Difference : BAG`Bag \times BAG`Bag \to BAG`Bag$;
.3 $Empty : () \to BAG`Bag$;
.4 $In : BAG`Elem \times BAG`Bag \to \mathbb{B}$;
.5 $Intersection : BAG`Bag \times BAG`Bag \to BAG`Bag$;
.6 $Join : BAG`Bag \times BAG`Bag \to BAG`Bag$;
.7 $Remove : BAG`Elem \times BAG`Bag \to BAG`Bag$;
.8 $SeqToBag : BAG`Elem^* \to BAG`Bag$;
.9 $Size : BAG`Bag \to \mathbb{N}$;
.10 $SubBag : BAG`Bag \times BAG`Bag \to \mathbb{B}$;
.11 $Union : BAG`Bag \times BAG`Bag \to BAG`Bag$

exports all

definitions

types

27.0 $Drink = BAG`Elem$;

28.0 $Cellar = BAG`Bag$;

29.0 $Bar = BAG`Bag$;

30.0 $Supplier = \mathsf{char}^*$;

31.0 $Pub = Cellar \times Bar$;

32.0 $BarLevel = BAG`Bag$;

33.0 $CellarLevel = BAG`Bag$;

34.0 $Stock = BAG`Bag$;

35.0 $Order = BAG`Bag$

functions

36.0 $BuyStock : Supplier \xrightarrow{m} Stock \times Supplier \times Order \times Pub \to Pub$

.1 $BuyStock\ (supps, s, stock, \mathsf{mk\text{-}}(c, r)) \triangleq$

.2 $\quad \mathsf{mk\text{-}}(BAG`Union\ (c, stock),$

.3 $\qquad r)$

.4 pre $s \in \mathsf{dom}\ supps \wedge$

.5 $\quad BAG`SubBag\ (stock, supps\ (s))$;

37.0 $RestockBar : Pub \times BarLevel \to Pub$

.1 $RestockBar\ (\mathsf{mk\text{-}}(c, r), bl) \triangleq$

.2 $\quad$ let $missing = BAG`Difference\ (bl, r)$ in

.3 $\quad$ let $can\text{-}restock = BAG`Intersection\ (missing, c)$ in

.4 $\quad \mathsf{mk\text{-}}(BAG`Difference\ (c, can\text{-}restock),$

.5 $\qquad BAG`Union\ (r, can\text{-}restock))$;

38.0 $Round : Drink^* \times Pub \to Pub$

.1 $Round\ (sold, \mathsf{mk\text{-}}(c, r)) \triangleq$

.2 $\quad \mathsf{mk\text{-}}(c,$

.3 $\qquad BAG`Difference\ (r, BAG`SeqToBag\ (sold)))$

.4 pre $BAG`SubBag\ (BAG`SeqToBag\ (sold), r)$;

39.0 $RestockCellar : CellarLevel \times Pub \times Supplier \xrightarrow{m} Stock \to Pub$

.1 $RestockCellar\ (cl, \mathsf{mk\text{-}}(c, r), sb) \triangleq$

.2 $\quad$ if $sb = \{\mapsto\}$

.3 $\quad$ then $\mathsf{mk\text{-}}(c, r)$

.4 $\quad$ else let $s \in \mathsf{dom}\ sb$ in

.5 $\qquad$ let $missing = BAG`Difference\ (cl, c)$ in

.6 $\qquad$ if $BAG`Size\ (missing) > 0$

.7 $\qquad$ then let $can\text{-}restock = BAG`Intersection\ (missing, sb\ (s))$ in

.8 $\qquad\quad RestockCellar\ (cl,$

.9 $\qquad\qquad\qquad \mathsf{mk\text{-}}(BAG`Union\ (c, can\text{-}restock), r),$

.10 $\qquad\qquad\qquad \{s\} \lhd sb)$

.11 $\qquad$ else $\mathsf{mk\text{-}}(c, r)$;

40.0 $Drink1 : Drink \times Pub \to Pub$

.1 $Drink1\ (dr, \mathsf{mk\text{-}}(c, r)) \triangleq$

.2 $\quad \mathsf{mk\text{-}}(c,$

.3 $\qquad BAG`Remove\ (dr, r))$

.4 pre $BAG`In\ (dr, r)$;

41.0 $Disaster : Pub \to \mathbb{B}$

.1 $Disaster\ (\mathsf{mk\text{-}}(c, r)) \triangleq$

.2 $\quad c = BAG`Empty\ () \wedge r = BAG`Empty\ ()$;

```
42.0  Unwanted : Drink × Pub → Pub
 .1   Unwanted (dr, mk-(c, r)) ≜
 .2     mk-(c,
 .3           BAG`Add (dr, r));

43.0  HighestStock : Supplier -m-> Stock → BAG`Bag
 .1   HighestStock (supps) ≜
 .2     if dom supps = {}
 .3     then BAG`Empty ()
 .4     else let s ∈ dom supps in
 .5          BAG`Join (supps (s), HighestStock ({s} ⩤ supps));

44.0  TotalDrinks : Pub → ℕ
 .1   TotalDrinks (mk-(c, r)) ≜
 .2     BAG`Size (c) + BAG`Size (r)

values

45.0  cellarlevel1 = {A ↦ 5, B ↦ 5, C ↦ 3};

46.0  barlevel1 = {A ↦ 2, B ↦ 2, C ↦ 5};

47.0  cellar1 = {A ↦ 8, B ↦ 5, C ↦ 4};

48.0  cellar2 = {B ↦ 1, C ↦ 4};

49.0  bar1 = {A ↦ 2, B ↦ 3, C ↦ 6};

50.0  bar2 = {A ↦ 3, C ↦ 2};

51.0  bar3 = {A ↦ 3, B ↦ 3};

52.0  pub1 = mk-(cellar1, bar1);

53.0  pub2 = mk-(cellar1, bar2);

54.0  pub3 = mk-(cellar2, bar1);

55.0  pub4 = mk-(cellar2, bar2);

56.0  pub5 = mk-(cellar1, bar3);

57.0  supps1 = {"Fizz" ↦ {A ↦ 10},
 .1             "Real" ↦ {B ↦ 10, C ↦ 2},
 .2             "Scrumpy" ↦ {B ↦ 1, C ↦ 10}};
```

58.0 $supps2 = \{$"$Fizz$" $\mapsto \{\mathrm{A} \mapsto 10\}$,
.1 "$Real$" $\mapsto \{\mathrm{B} \mapsto 1, \mathrm{C} \mapsto 5\}$,
.2 "$Scrumpy$" $\mapsto \{\mathrm{B} \mapsto 1, \mathrm{C} \mapsto 10\}\}$

end BAR