LECTURE 16: VENDING MACHINE CASE STUDY

Software Engineering Mike Wooldridge

1 Specification of a Vending Machine

- In this lecture, we will give a complete specification of a vending machine the sort you buy cans of coke or cigarettes from.
- First, we need to introduce some types; the first one will be *COIN*, representing all the coins that are accepted by the machine.

 $COIN == \{100, 50, 20, 10, 5, 2, 1\}$

- That is, there are coins in denominations of 100, 50, 20, 10, 5, 2, and 1 pence.
- We will also need a type for system messages
 - this is parachuted in:

[REPORT]

Mike Wooldridge

ecture 16 Software Engineeri	ing
• Next, we need a type <i>PROD</i> , representing all the products that the machine can sell.	-
[PROD]	
 We can define the state space of the vending machine thus: 	
<i>VendingMachine</i> <i>cost</i> : <i>PROD</i> → N <i>stock</i> : bag <i>PROD</i> <i>float</i> : bag <i>COIN</i>	
$\operatorname{dom} stock \subseteq \operatorname{dom} cost$	
ike Wooldridge	2

• The function *cost* return the cost of a product in pence. For example,

cost(MarsBar) = 25cost(Penguin) = 15

• The bag *stock* tells us how many items of each type are in stock. For example,

 $stock = \{Penguin \mapsto 2\}$

means that there are just 2 penguins in the machine.

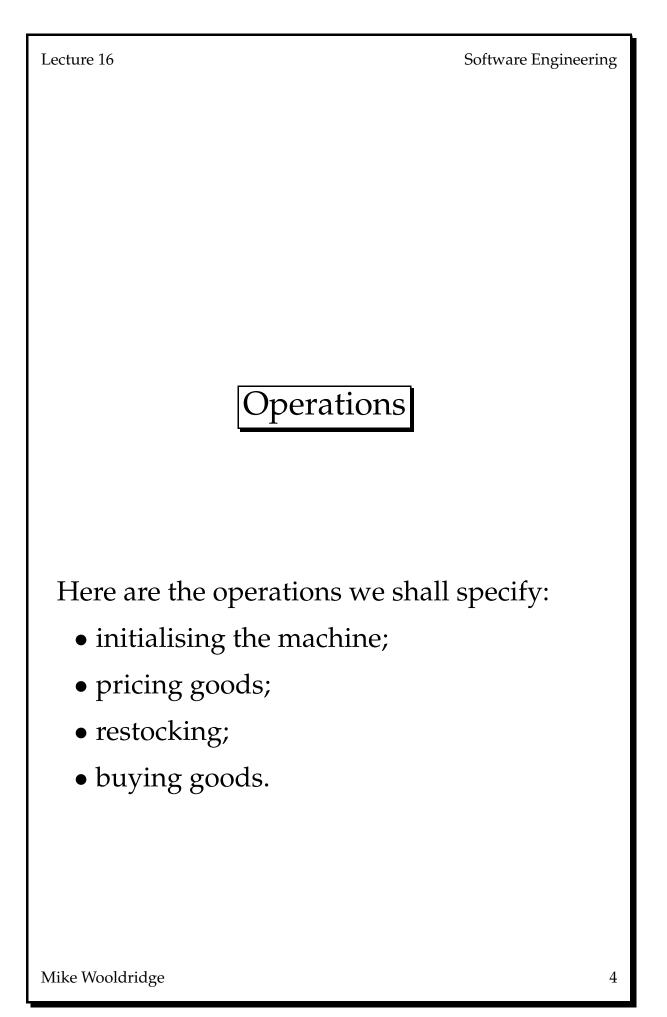
• The bag *float* records the coins that are currently in the machine; for example

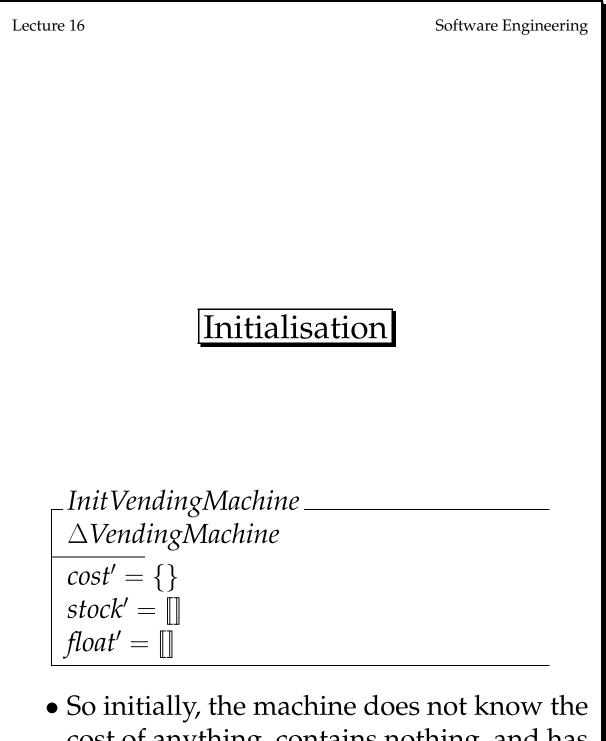
 $float = \{100 \mapsto 2, 50 \mapsto 8, 5 \mapsto 20\}$

means that there are $2 \times \pounds 1 \text{ coins}$, $8 \times 50 \text{ p}$ coins and $20 \times 5 \text{ p}$ coins.

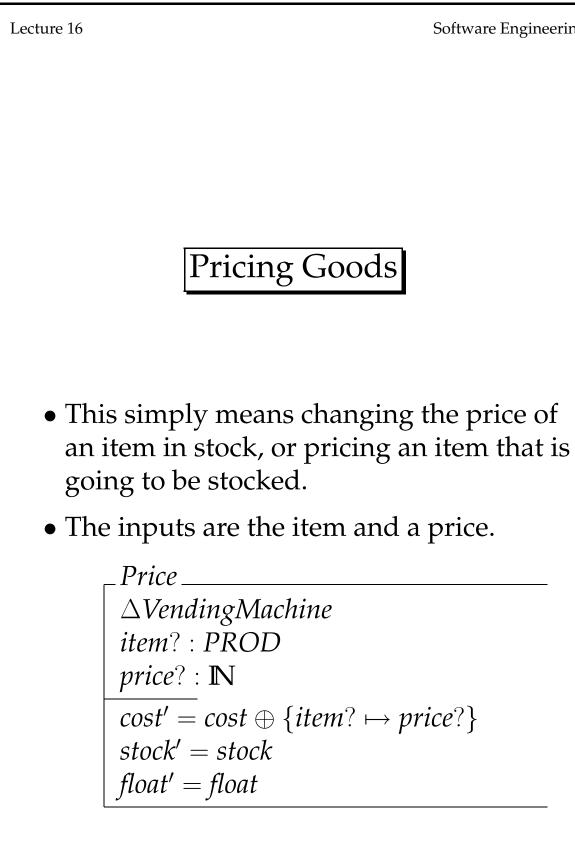
- QUESTION: Why are *stock* and *float* bags and not sets or sequences?
- The invariant dom *stock* ⊆ dom *cost* says that everything in the machine (i.e. in stock) must have a cost associated with it.

Mike Wooldridge





• So finitially, the machine does not know the cost of anything, contains nothing, and has no float.





Restocking

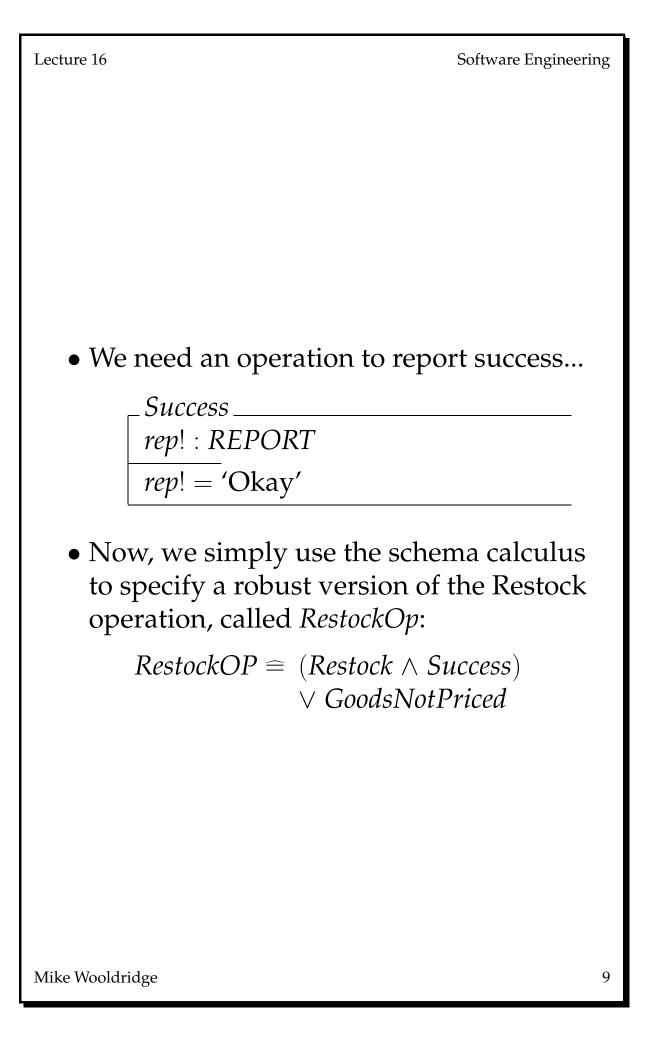
- The next operation to specify is that of restocking the machine with more goods.
- The only input is a new bag of products.
- The precondition dom *new*? ⊆ dom *cost* is implied by the invariant of *VendingMachine*'.

 $\begin{array}{l} Restock \\ \Delta Vending Machine \\ new? : bag PROD \\ \hline stock' = stock \uplus new? \\ float' = float \\ cost' = cost \end{array}$

• (Note that \uplus is the 'bag union' operator.)

Mike Wooldridge

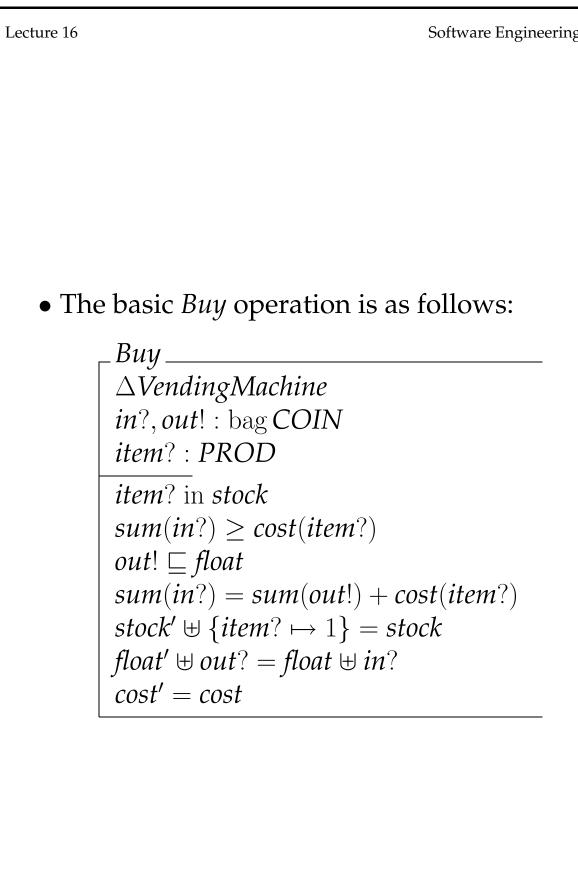
Lecture 16	Software Engineering
• We shall now make the open The <i>Restock</i> operation fails we attempt is made to add good not priced. We need a schem this situation.	vhen an ds which are
GoodsNotPriced EVendingMachine new? : bag PROD rep! : REPORT	
$\neg(\operatorname{dom} \operatorname{new}? \subseteq \operatorname{dom} \operatorname{cost})$	
rep! = 'Some goods are :	not priced'
Mike Wooldridge	8



```
Lecture 16
   • This schema expands to ...
             RestockOp _____
             \Delta Vending Machine
             new? : bag PROG
             rep! : REPORT
             \overline{cost'} = cost
            float' = float
             (stock' = stock \uplus new? \land
             rep! = 'Okay')
             (\neg(\operatorname{dom} \operatorname{new}? \subseteq \operatorname{dom} \operatorname{cost}) \land
              stock' = stock \land
              rep! = 'Some goods are not priced')
```

Lecture 16 Software Engineering Buying • The buying operation is a somewhat more complex operation ... • The inputs are the chosen item and some money. • We have to check that the item is in stock, and that the user has entered enough money to buy it. • We may also have to figure out what change to give ...

```
Software Engineering
Lecture 16
   • We assume that a function
     sum: bag COIN \rightarrow \mathbb{N}
     is available, which takes a bag of coins and
     calculates how much is in the bag. For
     example, given a bag containing 7 \times 2p,
     and 3 \times 5p coins,
          sum\{2 \mapsto 7, 5 \mapsto 3\} = (2 \times 7) + (5 \times 3)
                                  = 14 + 15
                                  = 29pence
Mike Wooldridge
                                                         12
```



- in? represents the coins entered; out! represents the change;
- item? is the item dispensed to the user;
- the 1st condition says that the item must be in stock;
- the 2nd condition says that the amount of money entered must be greater than or equal to the cost of the item;
- the 3rd condition says that the change given must have been part of the float;
- the 4th condition says that the the money entered must equal the change given plus the cost of the item;
- the 5th condition says that the stock before must be equal to the stock after, to which is added the dispensed item;
- the 6th condition says that the float after, together with the change dispensed must equal the float before plus the amount entered (i.e. no money disappears)