

A Save Area

A save area

A save area is a module with two operations: save and restore.

Packets of information are stored in a last-in first-out manner; in this respect, the module behaves as a stack.

Using Z

23-3

Using Z

23-4

Information

The structure of packets of information does not concern us at this level of abstraction:

[Record]

State

SaveArea _____

save-area : seq Record

InitSaveArea _____

SaveArea'

save-area' = {}

Using Z

23-5

Using Z

23-6

Question

Should this be a total operation?

<i>Save0</i> _____
<i>ΔSaveArea</i>
<i>record? : Record</i>
<i>status! : Status</i>
<i>save-area' = save-area ∪ {record?}</i>
<i>status! = ok</i>

SaveFullErr
$\exists \text{SaveArea}$
$\text{status}! : \text{Status}$
$\text{status}! = \text{full}$

$$\text{Save} \triangleq \text{Save}_0 \vee \text{SaveFullErr}$$

Restore_0
$\Delta \text{SaveArea}$
$r! : \text{Record}$
$\text{status}! : \text{Status}$
$\text{save_area} \neq \langle \rangle$
$\text{save_area} = \text{save_area}' \setminus \langle r' \rangle$
$\text{status}! = \text{ok}$

RestoreEmptyErr
$\exists \text{SaveArea}$
$\text{status}! : \text{Status}$
$\text{save_area} = \langle \rangle$
$\text{status}! = \text{empty}$

$$\text{Restore} \triangleq \text{Restore}_0 \vee \text{RestoreEmptyErr}$$

Preconditions for the save area

Design

Introduce two levels of memory: main and secondary.

Let n be the number of records that we can save in main memory:

$n : \mathbb{N}$
$n \geq 1$

Question

Save was defined nondeterministically; n has been defined loosely.

What is the difference?

$[X]$

$bseq : \mathbb{P}(\text{seq } X)$

$fseq : \mathbb{P}(\text{seq } X)$

$bseq = \{ s : \text{seq } X \mid \#s \leq n \}$

$fseq = \{ s : \text{seq } X \mid \#s = n \}$

CSaveArea

main : $bseq[\text{Record}]$

secondary : $\text{seq}(fseq[\text{Record}])$

Retrieve

SaveArea

CSaveArea

$\text{saveArea} = (\wedge / \text{secondary}) \wedge \text{main}$

Question

In the first specification, *saveArea* is initialised to the empty sequence. Can you calculate a suitable initialisation for our new level of design?

CSaveArea

record? : Record

status! : Status

Further design

The main memory storage will be implemented using an array and a counter:

CSaveArea

array : $\text{Array}[\text{Record}]$

count : $0 \dots n$

secondary : $\text{seq}(fseq[\text{Record}])$

where

$[X]$

Array : $\mathbb{P}(\mathbb{N} \leftrightarrow X)$

$\text{Array} = (1 \dots n) \rightarrow X$

Can you use the information from the state invariant to simplify your answer?

Question

What should Save_0 do now?

```

 $\Delta \text{CSaveArea}$ 
 $\text{record?} : \text{Record}$ 
 $\text{status!} : \text{Status}$ 
 $\text{main} = (1..count) \triangleleft array$ 

```

CCSaveArea

```

 $\exists \text{CSaveArea}$ 
 $\text{status!} : \text{Status}$ 
 $\text{status!} = \text{full}$ 

```

$\text{CCSaveFullErr} \triangleq \text{CCSave}_0 \vee \text{CCSaveFullErr}$

Refinement to code

We break the CCSave_0 operation into two disjuncts:

- CCUpdateMM : an operation that updates the main memory
- CCUpdateSM : an operation that updates the secondary memory

CCUpdateMM

```

 $\Delta \text{CSaveArea}$ 
 $\text{record?} : \text{Record}$ 
 $\text{status!} : \text{Status}$ 

```

```

 $\text{count} < n$ 
 $\text{count}' = \text{count} + 1$ 
 $\text{array}' = \text{array} \oplus \{\text{count} + 1 \leftarrow \text{record?}\}$ 
 $\text{secondary}' = \text{secondary} \cup \langle \text{array} \rangle$ 
 $\text{status!} = \text{ok}$ 

```

CCUpdateSM

```

 $\Delta \text{CSaveArea}$ 
 $\text{record?} : \text{Record}$ 
 $\text{status!} : \text{Status}$ 

```

```

 $\text{count} = n$ 
 $\text{count}' = 1$ 
 $\text{array}'[1] = \text{record?}$ 
 $\text{secondary}' = \text{secondary} \cap \langle \text{array} \rangle$ 
 $\text{status!} = \text{ok}$ 

```

Refinement

$$\text{save} \triangleq \text{CSaveArea1}, \text{status!} : [\text{true}, \text{CCSave}]$$

$$\text{Save} = \text{CSaveArea1}, \text{status!} : \left[\begin{array}{l} \text{CCUpdateMM} \\ \vee \\ \text{CCUpdateSM} \\ \vee \\ \text{true} , \text{CCSaveFullErr} \end{array} \right]$$

if $\text{count} < n \rightarrow$

$\text{CSaveArea1}, \text{status!} : [\text{count} < n, \text{CCUpdateMM}]$

[\bowtie]

$\square \text{ count} = n \rightarrow$

$\left[\begin{array}{l} \text{CCUpdateSM} \\ \vee \\ \text{[+]} \end{array} \right]$

$\text{CSaveArea1}, \text{status!} :$

$\left[\begin{array}{l} \text{count} = n , \text{CCSaveFullErr} \end{array} \right]$

fi

+

$\text{count}' = 1 \wedge$

$\text{array}'[1] = \text{record?} \wedge$

$\text{secondary}' =$

$\text{secondary}' \cap \langle \text{array} \rangle \wedge$

$\text{count}, \text{array},$

$\text{secondary}, \text{status!} :$

$\left[\begin{array}{l} \text{count}' = 1 \wedge \\ \text{array}'[1] = \text{record?} \wedge \\ \text{secondary}' = \\ \text{secondary}' \cap \langle \text{array} \rangle \wedge \\ \text{status!} = \text{ok} \\ \vee \\ \text{count}' = \text{count} \wedge \\ \text{array}' = \text{array} \wedge \\ \text{secondary}' = \text{secondary}' \wedge \\ \text{status!} = \text{full} \end{array} \right]$

```

if count < n →
  count, array, status! := count + 1, array ⊕ {count + 1 ↦ record!}, ok
  □ count = n →
    status!,  
 : [status! = ok ∧  
 secondary' = secondary ∩ ⟨array'⟩  
 ∨  
 secondary [true , status! = full ∧ secondary' = secondary]  
 ];
  if status! = ok →
    count, array := 1, array ⊕ {1 ↦ record!}
    □ status! = full →
      skip
    fi
  fi

```