Formal Development of a Real-Time Operating System Memory Manager in the Rodin Platform

Wen SU and Jean-Raymond Abrial

July 4, 2015

Contents

1 Contexts **2**

1.1 c00: Introducing the Size of the Memeory 2

1.2 c01: Mapping of Groups to Sizes 2

1.3 c03: Introducing the Call and Return for the Basic Operations 3

1.4 c04: More on the Mapping of Sizes to Groups 4

1.5 c05: A Two Dimensional Array for the Size of Groups 5

2 Machines **6**

2.1 m00: the Memory State and the Four Basic Operations 6

2.2 m01: Introducing Left and Removing Right 8

2.3 m02: Introducing Box for Block Size Group 10

2.4 m030: Introducing the Double Link (Forward Link Only) 12

2.5 m031: Introducing the Double Link (Backward Link) 15

2.6 m04: Removing Variable *sigma* and *box* 18

2.7 m05: Removing Varialbe $free$ and Introducing $free\_bit$ 20

2.8 m06: Introducing Allocating and Freeing Operations Calling the Basic Operations 22

2.9 m07: Removing Guards in Basic Operations 31

2.10 m08: Introducing Search (One Dimensional Array) 37

2.11 m09: Simplifying Search (Still One Dimensional Array) 44

2.12 m10: Introducing Two Dimensional Array for Search 51

Chapter 1

Contexts

1.1 c00: Introducing the Size of the Memeory

An Event-B Specification of c00
Creation Date: 4Jul2015 @ 10:10:16 PM

CONTEXT c00

Introducing the size of the memory

CONSTANTS

m

AXIOMS

`axm7` : $m > 0$

END

1.2 c01: Mapping of Groups to Sizes

An Event-B Specification of c01
Creation Date: 4Jul2015 @ 10:10:16 PM

CONTEXT c01

Introducing the groups and the mapping of groups to sizes

EXTENDS c00

CONSTANTS

d

g

AXIOMS

`axm1` : $d \in \mathbb{N}_1$

`axm2` : $g \in 1 \mathinner{..} m \twoheadrightarrow 1 \mathinner{..} d$

`axm3` : $d = g(m)$

END

1.3 c03: Introducing the Call and Return for the Basic Operations

An Event-B Specification of c03
Creation Date: 4Jul2015 @ 10:10:16 PM

CONTEXT c03
Introducing the call and return for the basic operations
EXTENDS c02
SETS

P

CONSTANTS

$call\_make\_free$
$return\_make\_free$
$call\_remove\_from\_free$
$return\_remove\_from\_free$
$call\_reduce\_create$
$return\_reduce\_create$
$call\_merge\_left$
$return\_merge\_left$
$call\_merge\_right$
$return\_merge\_right$
$undefined$

AXIOMS

axm1 : $partition(P, \{call\_make\_free\}, \{return\_make\_free\}, \{call\_remove\_from\_free\}, \{return\_remove\_from\_free\}, \{call\_reduce\_create\}, \{return\_reduce\_create\}, \{call\_merge\_left\}, \{return\_merge\_left\}, \{call\_merge\_right\}, \{return\_merge\_right\}, \{undefined\})$

END

1.4 c04: More on the Mapping of Sizes to Groups

An Event-B Specification of c04
Creation Date: 4Jul2015 @ 10:10:16 PM

CONTEXT c04
More on the mapping of sizes to groups
EXTENDS c03
CONSTANTS

$lower$
$upper$

g_srh

AXIOMS

axm2 : $lower \in 1 \mathinner{..} d \rightarrow 1 \mathinner{..} m$

axm3 : $upper \in 1 \mathinner{..} d \rightarrow 1 \mathinner{..} m$

axm4 : $\forall i \cdot i \in 1 \mathinner{..} d \Rightarrow g^{-1}[\{i\}] = lower(i) \mathinner{..} upper(i)$

axm5 : $\forall i \cdot i \in 1 \mathinner{..} d - 1 \Rightarrow lower(i+1) = upper(i) + 1$

axm6 : $lower(1) = 0$

axm7 : $upper(d) = m$

axm8 : $\forall q \cdot q \in 1 \mathinner{..} m \Rightarrow g(q) = min(\{i | i \in 1 \mathinner{..} d \land q \leq upper(i)\})$

axm9 : $\forall i, j \cdot i \in 1 \mathinner{..} d \land j \in 1 \mathinner{..} d \land i < j \Rightarrow lower(i) < lower(j)$

axm12 : $\forall q1, q2 \cdot q1 \in 1 \mathinner{..} m \land q2 \in 1 \mathinner{..} m \land q1 < q2 \Rightarrow g(q1) \leq g(q2)$

axm13 : $g\_srh \in 1 \mathinner{..} m \rightarrow 1 \mathinner{..} d$

axm14 : $\forall q \cdot q \in 1 \mathinner{..} lower(d) \Rightarrow g\_srh(q) = min(\{i | i \in 1 \mathinner{..} d \land q \leq lower(i)\})$

axm10 : $\forall q \cdot q \in 1 \mathinner{..} lower(d) \Rightarrow q \leq lower(g\_srh(q))$

axm11 : $\forall q, j \cdot q \in 1 \mathinner{..} lower(d) \land j \in 1 \mathinner{..} d \land g\_srh(q) < j \Rightarrow q < lower(j)$

END

1.5 c05: A Two Dimensional Array for the Size of Groups

An Event-B Specification of c05
Creation Date: 4Jul2015 @ 10:10:16 PM

CONTEXT c05
two dimensional array for the size of groups
EXTENDS c04
CONSTANTS

mf
ms
fl
ft
$search\_start$

AXIOMS

axm1 : $mf > 0$
axm2 : $ms > 0$
axm3 : $mf * ms = m + 1$
axm4 : $fl \in 1 \mathbin{..} m \to \mathbb{N}$
axm5 : $ft \in 1 \mathbin{..} m \to \mathbb{N}$
axm6 : $\forall q \cdot q \in 1 \mathbin{..} m \Rightarrow fl(q) * ms + ft(q) = q$
axm7 : $\forall q \cdot q \in 1 \mathbin{..} m \Rightarrow fl(q) \in 0 \mathbin{..} mf - 1$
axm8 : $\forall q \cdot q \in 1 \mathbin{..} m \Rightarrow ft(q) \in 0 \mathbin{..} ms - 1$
axm9 : $ms \geq 2$
axm19 : $\forall i, q \cdot i \in 1 \mathbin{..} d \land q \in 1 \mathbin{..} m \Rightarrow upper(g(q)) - lower(g(q)) = upper(i) - lower(i)$
axm10 : $search\_start \in 1 \mathbin{..} lower(d) \to 1 \mathbin{..} m$
axm11 : $\forall q \cdot q \in 1 \mathbin{..} lower(d) \Rightarrow search\_start(q) = q + (upper(g(q)) - lower(g(q)))$
thm1 : $\forall q \cdot q \in 1 \mathbin{..} lower(d) \Rightarrow g\_srh(q) = g(search\_start(q))$
// connect g(q) and g_srh(q)

axm20 : $\forall i \cdot i \in 0 \mathbin{..} mf - 1 \Rightarrow i * ms + ms - 1 = upper(g(i * ms + ms - 1))$
axm21 : $\forall i \cdot i \in 1 \mathbin{..} mf - 1 \Rightarrow i * ms = lower(g(i * ms))$
axm22 : $0 = lower(1)$
axm18 : $\forall q, k \cdot q \in 1 \mathbin{..} m \land$
$k \in fl(q) + 1 \mathbin{..} mf - 1$
$\Rightarrow$
$g((fl(q)+1)*ms) \mathbin{..} g(k*ms-1) = (\bigcup r \cdot r \in fl(q)+1 \mathbin{..} k-1 | g(r*ms) \mathbin{..} g(r*ms+ms-1))$
thm2 : $\forall q \cdot q \in 1 \mathbin{..} m \land \neg fl(q) = mf - 1$
$\Rightarrow$
$g((fl(q) + 1) * ms) \mathbin{..} d = (\bigcup r \cdot r \in fl(q) + 1 \mathbin{..} mf - 1 | g(r * ms) \mathbin{..} g(r * ms + ms - 1))$
axm17 : $\forall q1, q2, q3 \cdot q1 \in 1 \mathbin{..} m \land q2 \in 1 \mathbin{..} m \land q3 \in 1 \mathbin{..} m \land q1 < q2 \land q2 \leq q3$
$\Rightarrow$
$g(q1) \mathbin{..} g(q2 - 1) \cap g(q2) \mathbin{..} g(q3) = \varnothing$
thm3 : $\forall q \cdot q \in 1 \mathbin{..} m \land \neg fl(q) = mf - 1$
$\Rightarrow$
$g(q) \mathbin{..} d = g(q) \mathbin{..} g(fl(q) * ms + ms - 1) \cup g((fl(q) + 1) * ms) \mathbin{..} d$
thm4 : $\forall q \cdot q \in 1 \mathbin{..} m \land \neg fl(q) = mf - 1$
$\Rightarrow$
$g(q) \mathbin{..} g(fl(q) * ms + ms - 1) \cap g((fl(q) + 1) * ms) \mathbin{..} d = \varnothing$
axm16 : $\forall q1, q2, q3 \cdot q1 \in 1 \mathbin{..} m \land q2 \in 1 \mathbin{..} m \land q3 \in 1 \mathbin{..} m \land q1 < q2 \land q2 \leq q3$
$\Rightarrow$
$g(q1) \mathbin{..} g(q3) = g(q1) \mathbin{..} g(q2 - 1) \cup g(q2) \mathbin{..} g(q3)$

END

Chapter 2

Machines

2.1 m00: the Memory State and the Four Basic Operations

An Event-B Specification of m00
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m00
Introducing the memory state and the four basic operations
SEES c00
VARIABLES

size
right
free

INVARIANTS

inv1 : $size \in 0\,..\,m+1 \pfun 1\,..\,m$
inv2 : $\{0, m+1\} \subseteq dom(size)$
inv3 : $size(0) = 1$
inv4 : $size(m+1) = 1$
inv5 : $right \in 0\,..\,m \pfun 1\,..\,m+1$
inv6 : $dom(right) = dom(size) \setminus \{m+1\}$
inv7 : $\forall b \cdot b \in dom(right) \Rightarrow right(b) = b + size(b)$
inv8 : $free \subseteq dom(size)$
inv9 : $0 \notin free$
inv10 : $m+1 \notin free$
inv11 : $\forall b, c \cdot b \in dom(size) \land c \in dom(size) \land b \neq c \Rightarrow (c\,..\,c+size(c)-1) \cap (b\,..\,b+size(b)-1) = \varnothing$
inv14 : $\forall b, c \cdot b \in dom(right) \land c \in dom(right) \land right(b) = right(c) \Rightarrow b = c$
inv15 : $right \in 0\,..\,m \pinj 1\,..\,m+1$
inv16 : $\forall b \cdot b \in dom(right) \Rightarrow right(b) \neq b$
inv17 : $\forall b \cdot b \in dom(right) \land right(b) \in dom(right) \Rightarrow right(right(b)) \neq b$
inv13 : $\forall b \cdot b \in dom(right) \land right(b) \neq m+1 \Rightarrow right(b) \in dom(right)$
inv18 : $\forall b \cdot b \in dom(size) \land b \neq 0 \Rightarrow b \in ran(right)$
inv19 : $\forall b \cdot b \in dom(size) \land b \neq 0 \Rightarrow right^{-1}(b) \in dom(size)$
inv20 : $\forall b \cdot b \in dom(size) \land b \neq m+1 \Rightarrow right(b) \in dom(size)$
inv21 : $\forall b \cdot b \in dom(right^{-1}) \land right^{-1}(b) \in dom(right^{-1}) \Rightarrow right^{-1}(right^{-1}(b)) \neq b$
thm1 : $(\bigcup i \cdot i \in dom(size) | i\,..\,i+size(i)-1) = 0\,..\,m+1$

EVENTS
Initialisation
begin

```
            act1 : $free := \{1\}$
            act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
            act3 : $right := \{0 \mapsto 1, 1 \mapsto m+1\}$
    end
Event   $make\_free \;\widehat{=}$
    any
            $b$
    where
            grd1 : $b \in dom(size) \setminus free$
            grd2 : $b \notin \{0, m+1\}$
    then
            act1 : $free := free \cup \{b\}$
    end
Event   $remove\_from\_free \;\widehat{=}$
    any
            $b$
    where
            grd1 : $b \in free$
    then
            act1 : $free := free \setminus \{b\}$
    end
Event   $reduce\_create \;\widehat{=}$
    any
            $b$
            $q$
    where
            grd1 : $b \in dom(size) \setminus free$
            grd2 : $q < size(b)$
            grd3 : $q > 0$
            grd4 : $b \notin \{0, m+1\}$
    then
            act1 : $size := (\{b\} ⩤ size) \cup \{b \mapsto q\} \cup \{b+q \mapsto size(b) - q\}$
            act2 : $right := (\{b\} ⩤ right) \cup \{b \mapsto b+q\} \cup \{b+q \mapsto right(b)\}$
    end
Event   $merge\_right \;\widehat{=}$
    any
            $b$
    where
            grd1 : $b \in dom(size) \setminus free$
            grd2 : $b \notin \{0, m+1\}$
            grd3 : $right(b) \notin free$
            grd4 : $right(b) \notin \{0, m+1\}$
    then
            act1 : $size := (\{right(b), b\} ⩤ size) \cup \{b \mapsto size(b) + size(right(b))\}$
            act2 : $right := (\{right(b)\} ⩤ right ⩥ \{right(b)\}) \cup \{b \mapsto right(right(b))\}$
    end
END
```

2.2 m01: Introducing Left and Removing Right

An Event-B Specification of m01
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m01
Introducing left and removing right
REFINES m00
SEES c00
VARIABLES

$size$
$free$
$left$

INVARIANTS

inv1 : $left = right^{-1}$
inv2 : $\forall b \cdot b \in dom(right) \land b \neq 0 \Rightarrow b \in dom(left)$
inv3 : $\forall b \cdot b \in dom(size) \land b \in free \Rightarrow left(b) \notin free \land right(b) \notin free$

EVENTS
Initialisation
begin

act1 : $free := \{1\}$
act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$

end
Event *make_free* $\widehat{=}$
refines *make_free*
any

b

where

grd1 : $b \in dom(size) \setminus free$
grd2 : $b \notin \{0, m+1\}$
grd4 : $left^{-1}(b) \notin free$
grd3 : $left(b) \notin free$

then

act1 : $free := free \cup \{b\}$

end
Event *remove_from_free* $\widehat{=}$
extends *remove_from_free*
any

b

where

grd1 : $b \in free$

then

act1 : $free := free \setminus \{b\}$

end
Event *reduce_create* $\widehat{=}$
refines *reduce_create*
any

b
q

where

grd1 : $b \in dom(size) \setminus free$

grd2 : $q < \mathit{size}(b)$
grd3 : $q > 0$
grd4 : $b \notin \{0, m+1\}$

then

act1 : $\mathit{size} := (\{b\} \lhd \mathit{size}) \cup \{b \mapsto q\} \cup \{b+q \mapsto \mathit{size}(b) - q\}$
act3 : $\mathit{left} := (\{\mathit{left}^{-1}(b)\} \lhd \mathit{left}) \cup \{b+q \mapsto b\} \cup \{\mathit{left}^{-1}(b) \mapsto b+q\}$

end

Event *merge_right* $\widehat{=}$
refines *merge_right*

any

b

where

grd1 : $b \in \mathit{dom}(\mathit{size}) \setminus \mathit{free}$
grd2 : $b \notin \{0, m+1\}$
grd3 : $\mathit{left}^{-1}(b) \notin \mathit{free}$
grd4 : $\mathit{left}^{-1}(b) \notin \{0, m+1\}$

then

act1 : $\mathit{size} := (\{\mathit{left}^{-1}(b), b\} \lhd \mathit{size}) \cup \{b \mapsto \mathit{size}(b) + \mathit{size}(\mathit{left}^{-1}(b))\}$
act3 : $\mathit{left} := (\{\mathit{left}^{-1}(b)\} \lhd \mathit{left} \rhd \{\mathit{left}^{-1}(b)\}) \cup \{\mathit{left}^{-1}(\mathit{left}^{-1}(b)) \mapsto b\}$

end

END

2.3 m02: Introducing Box for Block Size Group

An Event-B Specification of m02
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m02
Introducing box
REFINES m01
SEES c01
VARIABLES

$size$
$free$
$left$
box

INVARIANTS

inv1 : $box \in free \rightarrow 1\,..\,d$
inv2 : $\forall b \cdot b \in free \Rightarrow box(b) = g(size(b))$

EVENTS
Initialisation
extended
begin

act1 : $free := \{1\}$
act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$
act5 : $box := \{1 \mapsto d\}$

end
Event *make_free* $\widehat{=}$
extends *make_free*
any

b

where

grd1 : $b \in dom(size) \setminus free$
grd2 : $b \notin \{0, m+1\}$
grd4 : $left^{-1}(b) \notin free$
grd3 : $left(b) \notin free$

then

act1 : $free := free \cup \{b\}$
act2 : $box(b) := g(size(b))$

end
Event *remove_from_free* $\widehat{=}$
extends *remove_from_free*
any

b

where

grd1 : $b \in free$

then

act1 : $free := free \setminus \{b\}$
act2 : $box := \{b\} \lhd box$

end
Event *reduce_create* $\widehat{=}$
extends *reduce_create*
any

b
q

where

grd1 : $b \in dom(size) \setminus free$
grd2 : $q < size(b)$
grd3 : $q > 0$
grd4 : $b \notin \{0, m+1\}$

then

act1 : $size := (\{b\} ⩤ size) \cup \{b \mapsto q\} \cup \{b+q \mapsto size(b) - q\}$
act3 : $left := (\{left^{-1}(b)\} ⩤ left) \cup \{b+q \mapsto b\} \cup \{left^{-1}(b) \mapsto b+q\}$

end

Event *merge_right* $\widehat{=}$
extends *merge_right*

any

b

where

grd1 : $b \in dom(size) \setminus free$
grd2 : $b \notin \{0, m+1\}$
grd3 : $left^{-1}(b) \notin free$
grd4 : $left^{-1}(b) \notin \{0, m+1\}$

then

act1 : $size := (\{left^{-1}(b), b\} ⩤ size) \cup \{b \mapsto size(b) + size(left^{-1}(b))\}$
act3 : $left := (\{left^{-1}(b)\} ⩤ left ⩥ \{left^{-1}(b)\}) \cup \{left^{-1}(left^{-1}(b)) \mapsto b\}$

end

END

2.4 m030: Introducing the Double Link (Forward Link Only)

An Event-B Specification of m030
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m030
Introducing the double link (nx only)
REFINES m02
SEES c01
VARIABLES

$size$
$free$
$left$
box
$sigma$
f
nx

INVARIANTS

inv1 : $sigma \in 1 \,..\, d \rightarrow \mathbb{P}(free)$
inv2 : $\forall i \cdot i \in 1 \,..\, d \Rightarrow sigma(i) = box^{-1}[\{i\}]$
inv3 : $f \in 1 \,..\, d \rightarrow \mathbb{Z}$
inv4 : $\forall i \cdot i \in 1 \,..\, d \Rightarrow f(i) \in sigma(i) \cup \{-1\}$
inv5 : $\forall i \cdot i \in 1 \,..\, d \land f(i) = -1 \Rightarrow sigma(i) = \varnothing$
inv6 : $nx \in (1 \,..\, d) \rightarrow (\mathbb{Z} \pfun \mathbb{Z})$
inv7 : $\forall i \cdot i \in 1 \,..\, d \Rightarrow nx(i) \in sigma(i) \bij (sigma(i) \cup \{-1\}) \setminus \{f(i)\}$
inv8 : $\forall i, p \cdot i \in 1 \,..\, d \land p \subseteq (nx(i))^{-1}[p] \Rightarrow p = \varnothing$

EVENTS
Initialisation
extended
begin

act1 : $free := \{1\}$
act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$
act5 : $box := \{1 \mapsto d\}$
act6 : $sigma := ((1 \,..\, d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1\}\}$
act7 : $f := ((1 \,..\, d-1) \times \{-1\}) \cup \{d \mapsto 1\}$
act8 : $nx := ((1 \,..\, d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$

end
Event *make_free* $\widehat{=}$
refines *make_free*
any

b

where

grd1 : $b \in dom(size) \setminus free$
grd2 : $b \notin \{0, m+1\}$
grd4 : $left^{-1}(b) \notin free$
grd3 : $left(b) \notin free$
grd5 : $g(size(b)) \in 1 \,..\, d$
grd6 : $b \notin sigma(g(size(b)))$
grd7 : $\forall p \cdot p \subseteq (nx(g(size(b))))^{-1}[p] \Rightarrow p = \varnothing$

then

act1 : $free := free \cup \{b\}$
act2 : $box(b) := g(size(b))$

act3 : $sigma(g(size(b))) := sigma(g(size(b))) \cup \{b\}$
act4 : $f(g(size(b))) := b$
act5 : $nx(g(size(b))) := nx(g(size(b))) \cup \{b \mapsto f(g(size(b)))\}$

end

Event *remove_from_free_1* $\widehat{=}$
refines *remove_from_free*

any

b

where

grd1 : $b \in free$
grd2 : $b \in sigma(g(size(b)))$
grd3 : $f(g(size(b))) \neq b$
grd4 : $\forall p \cdot p \subseteq ((nx(g(size(b))))^{-1}[p]) \Rightarrow p = \varnothing$
grd5 : $nx(g(size(b))) \in sigma(g(size(b))) ⤖ (sigma(g(size(b))) \cup \{-1\}) \setminus \{f(g(size(b)))\}$

then

act1 : $free := free \setminus \{b\}$
act2 : $box := \{b\} ⩤ box$
act3 : $sigma(g(size(b))) := sigma(g(size(b))) \setminus \{b\}$
act4 : $nx(g(size(b))) := (\{b\} ⩤ nx(g(size(b))) ⩥ \{b\}) \cup \{(nx(g(size(b))))^{-1}(b) \mapsto (nx(g(size(b))))(b)\}$

end

Event *remove_from_free_2* $\widehat{=}$
refines *remove_from_free*

any

b

where

grd1 : $b \in free$
grd2 : $b \in sigma(g(size(b)))$
grd3 : $f(g(size(b))) = b$
grd4 : $\forall p \cdot p \subseteq ((nx(g(size(b))))^{-1}[p]) \Rightarrow p = \varnothing$
grd5 : $nx(g(size(b))) \in sigma(g(size(b))) ⤖ (sigma(g(size(b))) \cup \{-1\}) \setminus \{b\}$
grd6 : $g(size(b)) \in 1 \mathrel{..} d$

then

act1 : $free := free \setminus \{b\}$
act2 : $box := \{b\} ⩤ box$
act3 : $sigma(g(size(b))) := sigma(g(size(b))) \setminus \{b\}$
act4 : $nx(g(size(b))) := \{b\} ⩤ nx(g(size(b)))$
act5 : $f(g(size(b))) := (nx(g(size(b))))(b)$

end

Event *reduce_create* $\widehat{=}$
extends *reduce_create*

any

b
q

where

grd1 : $b \in dom(size) \setminus free$
grd2 : $q < size(b)$
grd3 : $q > 0$
grd4 : $b \notin \{0, m+1\}$

then

act1 : $size := (\{b\} ⩤ size) \cup \{b \mapsto q\} \cup \{b + q \mapsto size(b) - q\}$
act3 : $left := (\{left^{-1}(b)\} ⩤ left) \cup \{b + q \mapsto b\} \cup \{left^{-1}(b) \mapsto b + q\}$

end

Event *merge_right* $\widehat{=}$

extends *merge_right*

  any

   b

  where

   grd1 : $b \in dom(size) \setminus free$

   grd2 : $b \notin \{0, m+1\}$

   grd3 : $left^{-1}(b) \notin free$

   grd4 : $left^{-1}(b) \notin \{0, m+1\}$

  then

   act1 : $size := (\{left^{-1}(b), b\} \lhd size) \cup \{b \mapsto size(b) + size(left^{-1}(b))\}$

   act3 : $left := (\{left^{-1}(b)\} \lhd left \rhd \{left^{-1}(b)\}) \cup \{left^{-1}(left^{-1}(b)) \mapsto b\}$

  end

END

2.5 m031: Introducing the Double Link (Backward Link)

An Event-B Specification of m031
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m031
Introducing the double link (pr now)
REFINES m030
SEES c01
VARIABLES

$size$
$free$
$left$
box
$sigma$
f
nx
pr

INVARIANTS

inv3 : $pr \in 1\,..\,d \rightarrow \mathbb{P}(\mathbb{Z} \times \mathbb{Z})$
inv2 : $\forall i \cdot i \in 1\,..\,d \Rightarrow pr(i) = \{-1\} \mathbin{\lhd\mkern-14mu-} (nx(i)^{-1} \cup \{f(i) \mapsto -1\})$

EVENTS
Initialisation
extended
begin

act1 : $free := \{1\}$
act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$
act5 : $box := \{1 \mapsto d\}$
act6 : $sigma := ((1\,..\,d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1\}\}$
act7 : $f := ((1\,..\,d-1) \times \{-1\}) \cup \{d \mapsto 1\}$
act8 : $nx := ((1\,..\,d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act9 : $pr := ((1\,..\,d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$

end
Event *make_free* $\widehat{=}$
refines *make_free*
any

b

where

grd1 : $b \in dom(size) \setminus free$
grd2 : $b \notin \{0, m+1\}$
grd4 : $left^{-1}(b) \notin free$
grd3 : $left(b) \notin free$
grd5 : $g(size(b)) \in 1\,..\,d$
grd6 : $b \notin sigma(g(size(b)))$
grd7 : $\forall p \cdot p \subseteq (nx(g(size(b))))^{-1}[p] \Rightarrow p = \varnothing$

then

act1 : $free := free \cup \{b\}$
act2 : $box(b) := g(size(b))$
act3 : $sigma(g(size(b))) := sigma(g(size(b))) \cup \{b\}$
act4 : $f(g(size(b))) := b$
act5 : $nx(g(size(b))) := nx(g(size(b))) \cup \{b \mapsto f(g(size(b)))\}$
act6 : $pr(g(size(b))) := (\{f(g(size(b)))\} \mathbin{\lhd\mkern-14mu-} pr(g(size(b)))) \cup (\{-1\} \mathbin{\lhd\mkern-14mu-} (\{f(g(size(b))) \mapsto b, b \mapsto -1\}))$

end

Event *remove_from_free_1* $\widehat{=}$
refines *remove_from_free_1*

any

b

where

grd1 : $b \in free$
grd2 : $b \in sigma(g(size(b)))$
grd3 : $f(g(size(b))) \neq b$
grd4 : $\forall p \cdot p \subseteq ((nx(g(size(b))))^{-1}[p]) \Rightarrow p = \varnothing$
grd5 : $nx(g(size(b))) \in sigma(g(size(b))) ⤖ (sigma(g(size(b))) \cup \{-1\}) \setminus \{f(g(size(b)))\}$

then

act1 : $free := free \setminus \{b\}$
act2 : $box := \{b\} ⩤ box$
act3 : $sigma(g(size(b))) := sigma(g(size(b))) \setminus \{b\}$
act4 : $nx(g(size(b))) := (\{b\} ⩤ nx(g(size(b))) ⩥ \{b\}) \cup \{(pr(g(size(b))))(b) \mapsto (nx(g(size(b))))(b)\}$
act5 : $pr(g(size(b))) := (\{b\} ⩤ pr(g(size(b))) ⩥ \{b\}) \cup (\{-1\} ⩤ \{nx(g(size(b)))(b) \mapsto pr(g(size(b)))(b)\})$

end

Event *remove_from_free_2* $\widehat{=}$
extends *remove_from_free_2*

any

b

where

grd1 : $b \in free$
grd2 : $b \in sigma(g(size(b)))$
grd3 : $f(g(size(b))) = b$
grd4 : $\forall p \cdot p \subseteq ((nx(g(size(b))))^{-1}[p]) \Rightarrow p = \varnothing$
grd5 : $nx(g(size(b))) \in sigma(g(size(b))) ⤖ (sigma(g(size(b))) \cup \{-1\}) \setminus \{b\}$
grd6 : $g(size(b)) \in 1 \,..\, d$

then

act1 : $free := free \setminus \{b\}$
act2 : $box := \{b\} ⩤ box$
act3 : $sigma(g(size(b))) := sigma(g(size(b))) \setminus \{b\}$
act4 : $nx(g(size(b))) := \{b\} ⩤ nx(g(size(b)))$
act5 : $f(g(size(b))) := (nx(g(size(b))))(b)$
act6 : $pr(g(size(b))) := (\{b, (nx(g(size(b))))(b)\} ⩤ pr(g(size(b)))) \cup (\{-1\} ⩤ \{(nx(g(size(b))))(b) \mapsto$ $(pr(g(size(b))))(b)\})$

end

Event *reduce_create* $\widehat{=}$
extends *reduce_create*

any

b
q

where

grd1 : $b \in dom(size) \setminus free$
grd2 : $q < size(b)$
grd3 : $q > 0$
grd4 : $b \notin \{0, m + 1\}$

then

act1 : $size := (\{b\} ⩤ size) \cup \{b \mapsto q\} \cup \{b + q \mapsto size(b) - q\}$
act3 : $left := (\{left^{-1}(b)\} ⩤ left) \cup \{b + q \mapsto b\} \cup \{left^{-1}(b) \mapsto b + q\}$

end

Event *merge_right* $\widehat{=}$

extends *merge_right*

any

b

where

grd1 : $b \in dom(size) \setminus free$

grd2 : $b \notin \{0, m+1\}$

grd3 : $left^{-1}(b) \notin free$

grd4 : $left^{-1}(b) \notin \{0, m+1\}$

then

act1 : $size := (\{left^{-1}(b), b\} \lhd size) \cup \{b \mapsto size(b) + size(left^{-1}(b))\}$

act3 : $left := (\{left^{-1}(b)\} \lhd left \rhd \{left^{-1}(b)\}) \cup \{left^{-1}(left^{-1}(b)) \mapsto b\}$

end

END

2.6 m04: Removing Variable *sigma* and *box*

An Event-B Specification of m04
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m04
Removing sigma and box
REFINES m031
SEES c01
VARIABLES

$size$
$free$
$left$
f
nx
pr

EVENTS
Initialisation
begin

act1 : $free := \{1\}$
act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$
act7 : $f := ((1 \,..\, d-1) \times \{-1\}) \cup \{d \mapsto 1\}$
act8 : $nx := ((1 \,..\, d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act9 : $pr := ((1 \,..\, d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$

end
Event *make_free* $\widehat{=}$
refines *make_free*
any

b

where

grd1 : $b \in dom(size) \setminus free$
grd2 : $b \notin \{0, m+1\}$
grd4 : $left^{-1}(b) \notin free$
grd3 : $left(b) \notin free$

then

act1 : $free := free \cup \{b\}$
act4 : $f(g(size(b))) := b$
act5 : $nx(g(size(b))) := nx(g(size(b))) \cup \{b \mapsto f(g(size(b)))\}$
act6 : $pr(g(size(b))) := (\{f(g(size(b)))\} ⩤ pr(g(size(b)))) \cup (\{-1\} ⩤ (\{f(g(size(b))) \mapsto b, b \mapsto -1\}))$

end
Event *remove_from_free_1* $\widehat{=}$
refines *remove_from_free_1*
any

b

where

grd1 : $b \in free$
grd3 : $f(g(size(b))) \neq b$

then

act1 : $free := free \setminus \{b\}$
act4 : $nx(g(size(b))) := (\{b\} ⩤ nx(g(size(b))) ⩥ \{b\}) \cup \{(pr(g(size(b))))(b) \mapsto (nx(g(size(b))))(b)\}$
act5 : $pr(g(size(b))) := (\{b\} ⩤ pr(g(size(b))) ⩥ \{b\}) \cup (\{-1\} ⩤ \{nx(g(size(b)))(b) \mapsto pr(g(size(b)))(b)\})$

end

Event $remove\_from\_free\_2 \;\widehat{=}$

refines $remove\_from\_free\_2$

any

b

where

grd1 : $b \in free$

grd3 : $f(g(size(b))) = b$

then

act1 : $free := free \setminus \{b\}$

act4 : $nx(g(size(b))) := \{b\} \ndres nx(g(size(b)))$

act5 : $f(g(size(b))) := (nx(g(size(b))))(b)$

act6 : $pr(g(size(b))) := (\{b, (nx(g(size(b))))(b)\} \ndres pr(g(size(b)))) \cup$
$(\{-1\} \ndres \{(nx(g(size(b))))(b) \mapsto (pr(g(size(b))))(b)\})$

end

Event $reduce\_create \;\widehat{=}$

extends $reduce\_create$

any

b
q

where

grd1 : $b \in dom(size) \setminus free$

grd2 : $q < size(b)$

grd3 : $q > 0$

grd4 : $b \notin \{0, m+1\}$

then

act1 : $size := (\{b\} \ndres size) \cup \{b \mapsto q\} \cup \{b + q \mapsto size(b) - q\}$

act3 : $left := (\{left^{-1}(b)\} \ndres left) \cup \{b + q \mapsto b\} \cup \{left^{-1}(b) \mapsto b + q\}$

end

Event $merge\_right \;\widehat{=}$

extends $merge\_right$

any

b

where

grd1 : $b \in dom(size) \setminus free$

grd2 : $b \notin \{0, m+1\}$

grd3 : $left^{-1}(b) \notin free$

grd4 : $left^{-1}(b) \notin \{0, m+1\}$

then

act1 : $size := (\{left^{-1}(b), b\} \ndres size) \cup \{b \mapsto size(b) + size(left^{-1}(b))\}$

act3 : $left := (\{left^{-1}(b)\} \ndres left \nrres \{left^{-1}(b)\}) \cup \{left^{-1}(left^{-1}(b)) \mapsto b\}$

end

END

2.7 m05: Removing Varialbe $free$ and Introducing $free\_bit$

An Event-B Specification of m05
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m05
Removing free and introducing free_bit
REFINES m04
SEES c01
VARIABLES

$size$
$left$
f
nx
pr
$free\_bit$

INVARIANTS

inv1 : $free\_bit \in dom(size) \rightarrow BOOL$
inv2 : $\forall b \cdot b \in dom(size) \Rightarrow free\_bit(b) = bool(b \in free)$

EVENTS
Initialisation
begin

act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$
act7 : $f := ((1\,..\,d-1) \times \{-1\}) \cup \{d \mapsto 1\}$
act8 : $nx := ((1\,..\,d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act9 : $pr := ((1\,..\,d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act10 : $free\_bit := \{0 \mapsto FALSE, 1 \mapsto TRUE, m+1 \mapsto FALSE\}$

end
Event $make\_free$ $\widehat{=}$
refines $make\_free$
any

b

where

grd5 : $b \in dom(size)$
grd2 : $b \notin \{0, m+1\}$
grd4 : $free\_bit(left^{-1}(b)) = FALSE$
grd3 : $free\_bit(left(b)) = FALSE$
grd6 : $free\_bit(b) = FALSE$

then

act1 : $free\_bit(b) := TRUE$
act4 : $f(g(size(b))) := b$
act5 : $nx(g(size(b))) := nx(g(size(b))) \cup \{b \mapsto f(g(size(b)))\}$
act6 : $pr(g(size(b))) := (\{f(g(size(b)))\} ⩤ pr(g(size(b)))) \cup (\{-1\} ⩤ (\{f(g(size(b))) \mapsto b, b \mapsto -1\}))$

end
Event $remove\_from\_free\_1$ $\widehat{=}$
refines $remove\_from\_free\_1$
any

b

where

grd1 : $b \in dom(size)$
grd4 : $free\_bit(b) = TRUE$
grd3 : $f(g(size(b))) \neq b$

then

act1 : $free\_bit(b) := FALSE$
act4 : $nx(g(size(b))) := (\{b\} \mathbin{⩤} nx(g(size(b))) \mathbin{⩥} \{b\}) \cup \{(pr(g(size(b))))(b) \mapsto (nx(g(size(b))))(b)\}$
act5 : $pr(g(size(b))) := (\{b\} \mathbin{⩤} pr(g(size(b))) \mathbin{⩥} \{b\}) \cup (\{-1\} \mathbin{⩤} \{nx(g(size(b)))(b) \mapsto pr(g(size(b)))(b)\})$

end

Event *remove_from_free_2* $\widehat{=}$
refines *remove_from_free_2*

any

b

where

grd1 : $b \in dom(size)$
grd4 : $free\_bit(b) = TRUE$
grd3 : $f(g(size(b))) = b$

then

act1 : $free\_bit(b) := FALSE$
act4 : $nx(g(size(b))) := \{b\} \mathbin{⩤} nx(g(size(b)))$
act5 : $f(g(size(b))) := (nx(g(size(b))))(b)$
act6 : $pr(g(size(b))) := (\{b, (nx(g(size(b))))(b)\} \mathbin{⩤} pr(g(size(b)))) \cup$
$(\{-1\} \mathbin{⩤} \{(nx(g(size(b))))(b) \mapsto (pr(g(size(b))))(b)\})$

end

Event *reduce_create* $\widehat{=}$
refines *reduce_create*

any

b
q

where

grd1 : $b \in dom(size)$
grd2 : $q < size(b)$
grd3 : $q > 0$
grd4 : $b \notin \{0, m + 1\}$
grd5 : $free\_bit(b) = FALSE$

then

act1 : $size := (\{b\} \mathbin{⩤} size) \cup \{b \mapsto q\} \cup \{b + q \mapsto size(b) - q\}$
act3 : $left := (\{left^{-1}(b)\} \mathbin{⩤} left) \cup \{b + q \mapsto b\} \cup \{left^{-1}(b) \mapsto b + q\}$
act4 : $free\_bit(b + q) := FALSE$

end

Event *merge_right* $\widehat{=}$
refines *merge_right*

any

b

where

grd1 : $b \in dom(size)$
grd2 : $b \notin \{0, m + 1\}$
grd3 : $free\_bit(left^{-1}(b)) = FALSE$
grd4 : $left^{-1}(b) \notin \{0, m + 1\}$
grd5 : $free\_bit(b) = FALSE$

then

act1 : $size := (\{left^{-1}(b), b\} \mathbin{⩤} size) \cup \{b \mapsto size(b) + size(left^{-1}(b))\}$
act3 : $left := (\{left^{-1}(b)\} \mathbin{⩤} left \mathbin{⩥} \{left^{-1}(b)\}) \cup \{left^{-1}(left^{-1}(b)) \mapsto b\}$
act4 : $free\_bit := \{left^{-1}(b)\} \mathbin{⩤} free\_bit$

end

END

2.8 m06: Introducing Allocating and Freeing Operations Calling the Basic Operations

An Event-B Specification of m06
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m06
Introducing allocating and freeing operations calling the basic operations
REFINES m05
SEES c03
VARIABLES

$size$
$left$
nx
pr
f
$free\_bit$
$prog$
adr
$b\_remove\_from\_free$
b_reduce_create
q_reduce_create
q_loc
b_make_free
$bloc$
b_merge_right

INVARIANTS

inv1 : $prog \in P$
inv2 : $adr \in \mathbb{N}$
inv3 : $adr = 0 \Rightarrow prog = undefined$
inv4 : $b\_remove\_from\_free \in \mathbb{N}$
inv5 : $b\_reduce\_create \in \mathbb{N}$
inv6 : $q\_reduce\_create \in \mathbb{N}$
inv7 : $q\_loc \in \mathbb{N}$
inv8 : $b\_make\_free \in \mathbb{N}$
inv9 : $prog = call\_remove\_from\_free$
$\Rightarrow$
$b\_remove\_from\_free \in dom(size) \wedge$
$free\_bit(b\_remove\_from\_free) = TRUE \wedge$
$(adr = 2 \Rightarrow q\_loc < size(b\_remove\_from\_free)) \wedge$
$(adr = 2 \Rightarrow q\_loc > 0)$
inv10 : $prog = return\_remove\_from\_free$
$\Rightarrow$
$b\_remove\_from\_free \in dom(size) \wedge$
$free\_bit(b\_remove\_from\_free) = FALSE \wedge$
$b\_remove\_from\_free \notin \{0, m + 1\} \wedge$
$free\_bit(left^{-1}(b\_remove\_from\_free)) = FALSE \wedge$
$(adr = 2 \Rightarrow q\_loc < size(b\_remove\_from\_free)) \wedge$
$(adr = 2 \Rightarrow q\_loc > 0)$

inv11 : $$\begin{array}{l} prog = call\_reduce\_create \\ \quad \Rightarrow \\ \quad q\_reduce\_create = q\_loc \land \\ \quad b\_reduce\_create \in dom(size) \land \\ \quad free\_bit(b\_reduce\_create) = FALSE \land \\ \quad q\_reduce\_create < size(b\_reduce\_create) \land \\ \quad q\_reduce\_create > 0 \land \\ \quad b\_reduce\_create \notin \{0, m+1\} \land \\ \quad free\_bit(left^{-1}(b\_reduce\_create)) = FALSE \end{array}$$

inv12 : $$\begin{array}{l} prog = return\_reduce\_create \\ \quad \Rightarrow \\ \quad q\_reduce\_create = q\_loc \land \\ \quad b\_reduce\_create \in dom(size) \land \\ \quad b\_reduce\_create + q\_reduce\_create \in dom(size) \setminus \{0, m+1\} \land \\ \quad left(b\_reduce\_create + q\_reduce\_create) = b\_reduce\_create \land \\ \quad free\_bit(b\_reduce\_create) = FALSE \land \\ \quad free\_bit(left^{-1}(b\_reduce\_create + q\_reduce\_create)) = FALSE \land \\ \quad free\_bit(b\_reduce\_create + q\_reduce\_create) = FALSE \end{array}$$

inv13 : $$\begin{array}{l} prog = call\_make\_free \\ \quad \Rightarrow \\ \quad b\_make\_free \in dom(size) \land \\ \quad b\_make\_free \notin \{0, m+1\} \land \\ \quad free\_bit(left^{-1}(b\_make\_free)) = FALSE \land \\ \quad free\_bit(left(b\_make\_free)) = FALSE \land \\ \quad free\_bit(b\_make\_free) = FALSE \end{array}$$

inv14 : $adr = 7 \Rightarrow prog = undefined$

inv15 : $adr = 10 \Rightarrow prog = undefined$

inv16 : $bloc \in \mathbb{N}$

inv17 : $b\_merge\_right \in \mathbb{N}$

inv18 : $$\begin{array}{l} prog = call\_remove\_from\_free \land \\ \quad adr = 5 \\ \quad \Rightarrow \\ \quad bloc \in dom(size) \land \\ \quad free\_bit(bloc) = FALSE \land \\ \quad bloc \notin \{0, m+1\} \land \\ \quad left(bloc) \notin \{0, m+1\} \land \\ \quad left(bloc) = b\_remove\_from\_free \land \\ \quad left(left(bloc)) \in dom(size) \end{array}$$

inv19 : $$\begin{array}{l} prog = return\_remove\_from\_free \land \\ \quad adr = 5 \\ \quad \Rightarrow \\ \quad bloc \in dom(size) \land \\ \quad free\_bit(bloc) = FALSE \land \\ \quad bloc \notin \{0, m+1\} \land \\ \quad left(bloc) \in dom(size) \land \\ \quad free\_bit(left(bloc)) = FALSE \land \\ \quad left(bloc) \notin \{0, m+1\} \land \\ \quad left(left(bloc)) \in dom(size) \land \\ \quad free\_bit(left(left(bloc))) = FALSE \end{array}$$

inv20 : $adr = 10$

$$\begin{array}{l}
\Rightarrow \\
bloc \in dom(size) \land \\
free\_bit(bloc) = FALSE \land \\
bloc \notin \{0, m+1\} \land \\
left^{-1}(bloc) \in dom(size) \land \\
free\_bit(left^{-1}(bloc)) = FALSE \land \\
left(bloc) \in dom(size) \land \\
free\_bit(left(bloc)) = FALSE
\end{array}$$

inv21 : $prog = return\_merge\_right \land$

$$\begin{array}{l}
adr = 9 \\
\Rightarrow \\
bloc \in dom(size) \land \\
free\_bit(bloc) = FALSE \land \\
bloc \notin \{0, m+1\} \land \\
left^{-1}(bloc) \in dom(size) \land \\
free\_bit(left^{-1}(bloc)) = FALSE \land \\
left(bloc) \in dom(size) \land \\
free\_bit(left(bloc)) = FALSE
\end{array}$$

inv22 : $adr = 7$

$$\begin{array}{l}
\Rightarrow \\
bloc \in dom(size) \land \\
free\_bit(bloc) = FALSE \land \\
bloc \notin \{0, m+1\} \land \\
left(bloc) \in dom(size) \land \\
free\_bit(left(bloc)) = FALSE
\end{array}$$

inv23 : $prog = return\_merge\_right \land$

$$\begin{array}{l}
adr = 6 \\
\Rightarrow \\
b\_merge\_right = bloc \land \\
bloc \in dom(size) \land \\
bloc \notin \{0, m+1\} \land \\
free\_bit(bloc) = FALSE \land \\
left(bloc) \in dom(size) \land \\
free\_bit(left(bloc)) = FALSE \land \\
left^{-1}(bloc) \in dom(size)
\end{array}$$

inv24 : $prog = call\_merge\_right \land$

$$\begin{array}{l}
adr = 6 \\
\Rightarrow \\
b\_merge\_right = bloc \land \\
bloc \in dom(size) \land \\
bloc \notin \{0, m+1\} \land \\
left^{-1}(bloc) \in dom(size) \land \\
free\_bit(left^{-1}(bloc)) = FALSE \land \\
left^{-1}(bloc) \notin \{0, m+1\} \land \\
free\_bit(bloc) = FALSE \land \\
left(bloc) \in dom(size) \land \\
free\_bit(left(bloc)) = FALSE
\end{array}$$

inv25 : $prog = call\_merge\_right \land$
$adr = 9$
$\Rightarrow$
$bloc = b\_merge\_right \land$
$bloc \in dom(size) \land$
$bloc \notin \{0, m+1\} \land$
$free\_bit(bloc) = FALSE \land$
$left^{-1}(bloc) \in dom(size) \land$
$free\_bit(left^{-1}(bloc)) = FALSE \land$
$left^{-1}(bloc) \notin \{0, m+1\} \land$
$left^{-1}(left^{-1}(bloc)) \in dom(size) \land$
$free\_bit(left^{-1}(left^{-1}(bloc))) = FALSE \land$
$bloc = b\_merge\_right \land$
$left(bloc) \in dom(size) \land$
$free\_bit(left(bloc)) = FALSE$

inv26 : $prog = return\_remove\_from\_free \land$
$adr = 8$
$\Rightarrow$
$bloc \in dom(size) \land$
$free\_bit(bloc) = FALSE \land$
$bloc \notin \{0, m+1\} \land$
$left^{-1}(bloc) \in dom(size) \land$
$free\_bit(left^{-1}(bloc)) = FALSE \land$
$left^{-1}(bloc) \notin \{0, m+1\} \land$
$left(bloc) \in dom(size) \land$
$free\_bit(left(bloc)) = FALSE \land$
$left^{-1}(left^{-1}(bloc)) \in dom(size) \land$
$free\_bit(left^{-1}(left^{-1}(bloc))) = FALSE$

inv27 : $prog = call\_remove\_from\_free \land$
$adr = 8$
$\Rightarrow$
$bloc \in dom(size) \land$
$free\_bit(bloc) = FALSE \land$
$bloc \notin \{0, m+1\} \land$
$left^{-1}(bloc) \in dom(size) \land$
$left^{-1}(bloc) \notin \{0, m+1\} \land$
$left^{-1}(bloc) = b\_remove\_from\_free \land$
$left(bloc) \in dom(size) \land$
$free\_bit(left(bloc)) = FALSE$

inv28 : $prog = call\_merge\_right \Rightarrow adr = 9 \lor adr = 6$

EVENTS

Initialisation

begin

act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$
act7 : $f := ((1 \mathinner{..} d-1) \times \{-1\}) \cup \{d \mapsto 1\}$
act8 : $nx := ((1 \mathinner{..} d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act9 : $pr := ((1 \mathinner{..} d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act10 : $free\_bit := \{0 \mapsto FALSE, 1 \mapsto TRUE, m+1 \mapsto FALSE\}$
act11 : $prog := undefined$
act12 : $adr := 0$
act13 : $b\_remove\_from\_free := 0$
act14 : $b\_reduce\_create := 0$
act15 : $q\_reduce\_create := 0$
act16 : $q\_loc := 0$
act17 : $b\_make\_free := 0$
act18 : $bloc := 0$

act19 : $b\_merge\_right := 0$

end

Event *allocate_1_1* $\widehat{=}$

any

b

q

where

grd1 : $b \in dom(size)$

grd2 : $free\_bit(b) = TRUE$

grd3 : $q \in 1 \mathinner{..} m$

grd4 : $q < size(b)$

grd5 : $q > 0$

grd6 : $adr = 0$

then

act1 : $adr := 2$

act2 : $prog := call\_remove\_from\_free$

act3 : $b\_remove\_from\_free := b$

act4 : $q\_loc := q$

end

Event *allocate_1_2* $\widehat{=}$

when

grd1 : $prog = return\_remove\_from\_free$

grd2 : $adr = 2$

then

act1 : $adr := 3$

act2 : $prog := call\_reduce\_create$

act3 : $b\_reduce\_create := b\_remove\_from\_free$

act4 : $q\_reduce\_create := q\_loc$

end

Event *allocate_1_3* $\widehat{=}$

when

grd1 : $prog = return\_reduce\_create$

grd2 : $adr = 3$

then

act1 : $adr := 4$

act2 : $prog := call\_make\_free$

act3 : $b\_make\_free := b\_reduce\_create + q\_loc$

end

Event *allocate_1_4* $\widehat{=}$

when

grd1 : $prog = return\_make\_free$

grd2 : $adr = 4$

then

act1 : $adr := 0$

act2 : $prog := undefined$

end

Event *allocate_2_1* $\widehat{=}$

any

b

q

where

grd1 : $b \in dom(size)$

```
        grd2 : $free\_bit(b) = TRUE$
        grd3 : $q = size(b)$
        grd4 : $adr = 0$
    then
        act1 : $adr := 1$
        act2 : $prog := call\_remove\_from\_free$
        act3 : $b\_remove\_from\_free := b$
    end
Event  allocate_2_2 $\widehat{=}$
    when
        grd1 : $prog = return\_remove\_from\_free$
        grd2 : $adr = 1$
    then
        act1 : $prog := undefined$
        act2 : $adr := 0$
    end
Event  free_1_1 $\widehat{=}$
    any
        $b$
    where
        grd1 : $b \in dom(size)$
        grd2 : $free\_bit(b) = FALSE$
        grd3 : $b \notin \{0, m + 1\}$
        grd4 : $left(b) \in dom(size)$
        grd5 : $free\_bit(left(b)) = TRUE$
        grd6 : $adr = 0$
    then
        act1 : $adr := 5$
        act2 : $prog := call\_remove\_from\_free$
        act3 : $b\_remove\_from\_free := left(b)$
        act4 : $bloc := b$
    end
Event  free_1_2 $\widehat{=}$
    when
        grd1 : $prog = return\_remove\_from\_free$
        grd2 : $adr = 5$
    then
        act1 : $adr := 6$
        act2 : $prog := call\_merge\_right$
        act3 : $b\_merge\_right := left(bloc)$
        act4 : $bloc := left(bloc)$
    end
Event  free_1_3 $\widehat{=}$
    when
        grd1 : $adr = 6$
        grd2 : $prog = return\_merge\_right$
    then
        act1 : $adr := 7$
        act2 : $prog := undefined$
    end
Event  free_2 $\widehat{=}$
    any
    b
  where
      grd1 : $b \in dom(size)$
      grd2 : $free\_bit(b) = FALSE$
      grd3 : $b \notin \{0, m+1\}$
      grd4 : $left(b) \in dom(size)$
      grd5 : $free\_bit(left(b)) = FALSE$
      grd6 : $adr = 0$
  then
      act1 : $adr := 7$
      act2 : $bloc := b$
  end
Event  free_3_1 ≙
  when
      grd1 : $adr = 7$
      grd2 : $free\_bit(left^{-1}(bloc)) = TRUE$
  then
      act1 : $adr := 8$
      act2 : $prog := call\_remove\_from\_free$
      act3 : $b\_remove\_from\_free := left^{-1}(bloc)$
  end
Event  free_3_2 ≙
  when
      grd1 : $adr = 8$
      grd2 : $prog = return\_remove\_from\_free$
  then
      act1 : $adr := 9$
      act2 : $prog := call\_merge\_right$
      act3 : $b\_merge\_right := bloc$
  end
Event  free_3_3 ≙
  when
      grd1 : $adr = 9$
      grd2 : $prog = return\_merge\_right$
  then
      act1 : $adr := 10$
      act2 : $prog := undefined$
  end
Event  free_4 ≙
  when
      grd1 : $adr = 7$
      grd2 : $left^{-1}(bloc) \in dom(size)$
      grd3 : $free\_bit(left^{-1}(bloc)) = FALSE$
  then
      act1 : $adr := 10$
  end
Event  free_5 ≙
  when
      grd1 : $adr = 10$
  then
      act1 : $adr := 11$
```

act2 : $prog := call\_make\_free$

act3 : $b\_make\_free := bloc$

end

Event *free_6* $\widehat{=}$

when

grd1 : $adr = 11$

grd2 : $prog = return\_make\_free$

then

act1 : $adr := 0$

act2 : $prog := undefined$

end

Event *make_free* $\widehat{=}$
refines *make_free*

when

grd6 : $prog = call\_make\_free$

with

b : $b = b\_make\_free$

then

act1 : $free\_bit(b\_make\_free) := TRUE$

act6 : $f(g(size(b\_make\_free))) := b\_make\_free$

act3 : $nx(g(size(b\_make\_free))) := nx(g(size(b\_make\_free))) \cup \{b\_make\_free \mapsto f(g(size(b\_make\_free)))\}$

act5 : $pr(g(size(b\_make\_free))) := (\{f(g(size(b\_make\_free)))\} ⩤ pr(g(size(b\_make\_free)))) \cup (\{-1\} ⩤ (\{f(g(size(b\_make\_free))) \mapsto b\_make\_free, b\_make\_free \mapsto -1\}))$

act4 : $prog := return\_make\_free$

end

Event *remove_from_free_1* $\widehat{=}$
refines *remove_from_free_1*

when

grd3 : $prog = call\_remove\_from\_free$

grd2 : $free\_bit(b\_remove\_from\_free) = TRUE$
free_bit(b_remove_from_free) = TRUE

grd1 : $b\_remove\_from\_free \in dom(size)$
b_remove_from_free ∈ dom(size)

grd4 : $f(g(size(b\_remove\_from\_free))) \neq b\_remove\_from\_free$

with

b : $b = b\_remove\_from\_free$

then

act1 : $free\_bit(b\_remove\_from\_free) := FALSE$

act3 : $nx(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} ⩤ nx(g(size(b\_remove\_from\_free))) ⩥ \{b\_remove\_from\_free\}) \cup \{(pr(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free) \mapsto (nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)\}$

act5 : $pr(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} ⩤ pr(g(size(b\_remove\_from\_free))) ⩥ \{b\_remove\_from\_free\}) \cup (\{-1\} ⩤ \{nx(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free) \mapsto pr(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free)\})$

act4 : $prog := return\_remove\_from\_free$

end

Event *remove_from_free_2* $\widehat{=}$
refines *remove_from_free_2*

when

grd1 : $prog = call\_remove\_from\_free$

grd2 : $\mathit{free\_bit}(\mathit{b\_remove\_from\_free}) = TRUE$
grd3 : $\mathit{b\_remove\_from\_free} \in \mathrm{dom}(\mathit{size})$
grd4 : $f(g(\mathit{size}(\mathit{b\_remove\_from\_free}))) = \mathit{b\_remove\_from\_free}$

with

b : $b = \mathit{b\_remove\_from\_free}$

then

act1 : $\mathit{free\_bit}(\mathit{b\_remove\_from\_free}) := FALSE$
act2 : $nx(g(\mathit{size}(\mathit{b\_remove\_from\_free}))) := \{\mathit{b\_remove\_from\_free}\} \mathbin{\lhd\!\!\!-} nx(g(\mathit{size}(\mathit{b\_remove\_from\_free})))$
act3 : $f(g(\mathit{size}(\mathit{b\_remove\_from\_free}))) := (nx(g(\mathit{size}(\mathit{b\_remove\_from\_free}))))(\mathit{b\_remove\_from\_free})$
act4 : $pr(g(\mathit{size}(\mathit{b\_remove\_from\_free}))) := (\{\mathit{b\_remove\_from\_free},$
$(nx(g(\mathit{size}(\mathit{b\_remove\_from\_free}))))(\mathit{b\_remove\_from\_free})\} \mathbin{\lhd\!\!\!-}$
$pr(g(\mathit{size}(\mathit{b\_remove\_from\_free})))) \cup (\{-1\} \mathbin{\lhd\!\!\!-}$
$\{(nx(g(\mathit{size}(\mathit{b\_remove\_from\_free}))))(\mathit{b\_remove\_from\_free}) \mapsto$
$(pr(g(\mathit{size}(\mathit{b\_remove\_from\_free}))))(\mathit{b\_remove\_from\_free})\})$
act5 : $\mathit{prog} := \mathit{return\_remove\_from\_free}$

end

Event *reduce_create* $\widehat{=}$
refines *reduce_create*

when

grd6 : $\mathit{prog} = \mathit{call\_reduce\_create}$

with

b : $b = \mathit{b\_reduce\_create}$
q : $q = \mathit{q\_reduce\_create}$

then

act1 : $\mathit{size} := (\{\mathit{b\_reduce\_create}\} \mathbin{\lhd\!\!\!-} \mathit{size}) \cup \{\mathit{b\_reduce\_create} \mapsto \mathit{q\_reduce\_create}\} \cup$
$\{\mathit{b\_reduce\_create} + \mathit{q\_reduce\_create} \mapsto \mathit{size}(\mathit{b\_reduce\_create}) - \mathit{q\_reduce\_create}\}$
act3 : $\mathit{left} := (\{\mathit{left}^{-1}(\mathit{b\_reduce\_create})\} \mathbin{\lhd\!\!\!-} \mathit{left}) \cup \{\mathit{b\_reduce\_create} + \mathit{q\_reduce\_create} \mapsto \mathit{b\_reduce\_create}\} \cup$
$\{\mathit{left}^{-1}(\mathit{b\_reduce\_create}) \mapsto \mathit{b\_reduce\_create} + \mathit{q\_reduce\_create}\}$
act4 : $\mathit{free\_bit}(\mathit{b\_reduce\_create} + \mathit{q\_reduce\_create}) := FALSE$
act5 : $\mathit{prog} := \mathit{return\_reduce\_create}$

end

Event *merge_right* $\widehat{=}$
refines *merge_right*

when

grd6 : $\mathit{prog} = \mathit{call\_merge\_right}$
grd1 : $\mathit{b\_merge\_right} \in \mathrm{dom}(\mathit{size})$
grd2 : $\mathit{b\_merge\_right} \notin \{0, m+1\}$
grd4 : $\mathit{left}^{-1}(\mathit{b\_merge\_right}) \notin \{0, m+1\}$
grd3 : $\mathit{free\_bit}(\mathit{left}^{-1}(\mathit{b\_merge\_right})) = FALSE$
grd5 : $\mathit{free\_bit}(\mathit{b\_merge\_right}) = FALSE$

with

b : $b = \mathit{b\_merge\_right}$

then

act1 : $\mathit{size} := (\{\mathit{left}^{-1}(\mathit{b\_merge\_right}), \mathit{b\_merge\_right}\} \mathbin{\lhd\!\!\!-} \mathit{size}) \cup$
$\{\mathit{b\_merge\_right} \mapsto \mathit{size}(\mathit{b\_merge\_right}) + \mathit{size}(\mathit{left}^{-1}(\mathit{b\_merge\_right}))\}$
act3 : $\mathit{left} := (\{\mathit{left}^{-1}(\mathit{b\_merge\_right})\} \mathbin{\lhd\!\!\!-} \mathit{left} \mathbin{-\!\!\!\rhd} \{\mathit{left}^{-1}(\mathit{b\_merge\_right})\}) \cup$
$\{\mathit{left}^{-1}(\mathit{left}^{-1}(\mathit{b\_merge\_right})) \mapsto \mathit{b\_merge\_right}\}$
act4 : $\mathit{free\_bit} := \{\mathit{left}^{-1}(\mathit{b\_merge\_right})\} \mathbin{\lhd\!\!\!-} \mathit{free\_bit}$
act5 : $\mathit{prog} := \mathit{return\_merge\_right}$

end

END

2.9 m07: Removing Guards in Basic Operations

An Event-B Specification of m07
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m07
Removing guards in basic operations
REFINES m06
SEES c03
VARIABLES

$size$
$left$
nx
pr
f
$free\_bit$
$prog$
adr
$b\_remove\_from\_free$
b_reduce_create
q_reduce_create
q_loc
b_make_free
$bloc$
b_merge_right

EVENTS
Initialisation
begin

act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$
act7 : $f := ((1 \mathrel{..} d-1) \times \{-1\}) \cup \{d \mapsto 1\}$
act8 : $nx := ((1 \mathrel{..} d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act9 : $pr := ((1 \mathrel{..} d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act10 : $free\_bit := \{0 \mapsto FALSE, 1 \mapsto TRUE, m+1 \mapsto FALSE\}$
act11 : $prog := undefined$
act12 : $adr := 0$
act13 : $b\_remove\_from\_free := 0$
act14 : $b\_reduce\_create := 0$
act15 : $q\_reduce\_create := 0$
act16 : $q\_loc := 0$
act17 : $b\_make\_free := 0$
act18 : $bloc := 0$
act19 : $b\_merge\_right := 0$

end
Event $allocate\_1\_1 \widehat{=}$
extends $allocate\_1\_1$
any

b
q

where

grd1 : $b \in dom(size)$
grd2 : $free\_bit(b) = TRUE$
grd3 : $q \in 1 \mathrel{..} m$
grd4 : $q < size(b)$

grd5 : $q > 0$

grd6 : $adr = 0$

then

act1 : $adr := 2$

act2 : $prog := call\_remove\_from\_free$

act3 : $b\_remove\_from\_free := b$

act4 : $q\_loc := q$

end

Event *allocate_1_2* $\widehat{=}$

extends *allocate_1_2*

when

grd1 : $prog = return\_remove\_from\_free$

grd2 : $adr = 2$

then

act1 : $adr := 3$

act2 : $prog := call\_reduce\_create$

act3 : $b\_reduce\_create := b\_remove\_from\_free$

act4 : $q\_reduce\_create := q\_loc$

end

Event *allocate_1_3* $\widehat{=}$

extends *allocate_1_3*

when

grd1 : $prog = return\_reduce\_create$

grd2 : $adr = 3$

then

act1 : $adr := 4$

act2 : $prog := call\_make\_free$

act3 : $b\_make\_free := b\_reduce\_create + q\_loc$

end

Event *allocate_1_4* $\widehat{=}$

extends *allocate_1_4*

when

grd1 : $prog = return\_make\_free$

grd2 : $adr = 4$

then

act1 : $adr := 0$

act2 : $prog := undefined$

end

Event *allocate_2_1* $\widehat{=}$

extends *allocate_2_1*

any

b

q

where

grd1 : $b \in dom(size)$

grd2 : $free\_bit(b) = TRUE$

grd3 : $q = size(b)$

grd4 : $adr = 0$

then

act1 : $adr := 1$

act2 : $prog := call\_remove\_from\_free$

act3 : $b\_remove\_from\_free := b$

end

Event *allocate_2_2* $\widehat{=}$
extends *allocate_2_2*
when
grd1 : $prog = return\_remove\_from\_free$
grd2 : $adr = 1$
then
act1 : $prog := undefined$
act2 : $adr := 0$
end
Event *free_1_1* $\widehat{=}$
extends *free_1_1*
any
b
where
grd1 : $b \in dom(size)$
grd2 : $free\_bit(b) = FALSE$
grd3 : $b \notin \{0, m+1\}$
grd4 : $left(b) \in dom(size)$
grd5 : $free\_bit(left(b)) = TRUE$
grd6 : $adr = 0$
then
act1 : $adr := 5$
act2 : $prog := call\_remove\_from\_free$
act3 : $b\_remove\_from\_free := left(b)$
act4 : $bloc := b$
end
Event *free_1_2* $\widehat{=}$
extends *free_1_2*
when
grd1 : $prog = return\_remove\_from\_free$
grd2 : $adr = 5$
then
act1 : $adr := 6$
act2 : $prog := call\_merge\_right$
act3 : $b\_merge\_right := left(bloc)$
act4 : $bloc := left(bloc)$
end
Event *free_1_3* $\widehat{=}$
extends *free_1_3*
when
grd1 : $adr = 6$
grd2 : $prog = return\_merge\_right$
then
act1 : $adr := 7$
act2 : $prog := undefined$
end
Event *free_2* $\widehat{=}$
extends *free_2*
any
b
where
grd1 : $b \in dom(size)$
grd2 : $free\_bit(b) = FALSE$

`grd3` : $b \notin \{0, m+1\}$
`grd4` : $left(b) \in dom(size)$
`grd5` : $free\_bit(left(b)) = FALSE$
`grd6` : $adr = 0$

then

`act1` : $adr := 7$
`act2` : $bloc := b$

end

Event *free_3_1* $\widehat{=}$
refines *free_3_1*

when

`grd1` : $adr = 7$
grd3 : $left^{-1}(bloc) = bloc + size(bloc)$
`grd2` : $free\_bit(bloc + size(bloc)) = TRUE$

then

`act1` : $adr := 8$
`act2` : $prog := call\_remove\_from\_free$
`act3` : $b\_remove\_from\_free := bloc + size(bloc)$

end

Event *free_3_2* $\widehat{=}$
extends *free_3_2*

when

`grd1` : $adr = 8$
`grd2` : $prog = return\_remove\_from\_free$

then

`act1` : $adr := 9$
`act2` : $prog := call\_merge\_right$
`act3` : $b\_merge\_right := bloc$

end

Event *free_3_3* $\widehat{=}$
extends *free_3_3*

when

`grd1` : $adr = 9$
`grd2` : $prog = return\_merge\_right$

then

`act1` : $adr := 10$
`act2` : $prog := undefined$

end

Event *free_4* $\widehat{=}$
refines *free_4*

when

`grd1` : $adr = 7$
grd4 : $left^{-1}(bloc) = bloc + size(bloc)$
grd2 : $bloc + size(bloc) \in dom(size)$
`grd3` : $free\_bit(bloc + size(bloc)) = FALSE$

then

`act1` : $adr := 10$

end

Event *free_5* $\widehat{=}$
extends *free_5*

when

`grd1` : $adr = 10$

then

act1 : $adr := 11$

act2 : $prog := call\_make\_free$

act3 : $b\_make\_free := bloc$

end

Event *free_6* $\widehat{=}$

extends *free_6*

when

grd1 : $adr = 11$

grd2 : $prog = return\_make\_free$

then

act1 : $adr := 0$

act2 : $prog := undefined$

end

Event *make_free* $\widehat{=}$

extends *make_free*

when

grd6 : $prog = call\_make\_free$

then

act1 : $free\_bit(b\_make\_free) := TRUE$

act6 : $f(g(size(b\_make\_free))) := b\_make\_free$

act3 : $nx(g(size(b\_make\_free))) := nx(g(size(b\_make\_free))) \cup$
$\{b\_make\_free \mapsto f(g(size(b\_make\_free)))\}$

act5 : $pr(g(size(b\_make\_free))) := (\{f(g(size(b\_make\_free)))\} \mathbin{⩤} pr(g(size(b\_make\_free)))) \cup$
$(\{-1\} \mathbin{⩤} (\{f(g(size(b\_make\_free))) \mapsto b\_make\_free, b\_make\_free \mapsto -1\}))$

act4 : $prog := return\_make\_free$

end

Event *remove_from_free_1* $\widehat{=}$

refines *remove_from_free_1*

when

grd3 : $prog = call\_remove\_from\_free$

grd4 : $f(g(size(b\_remove\_from\_free))) \neq b\_remove\_from\_free$

then

act1 : $free\_bit(b\_remove\_from\_free) := FALSE$

act3 : $nx(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} \mathbin{⩤}$
$nx(g(size(b\_remove\_from\_free))) \mathbin{⩥} \{b\_remove\_from\_free\}) \cup$
$\{(pr(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free) \mapsto$
$nx(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free)\}$

act4 : $prog := return\_remove\_from\_free$

act5 : $pr(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} \mathbin{⩤}$
$pr(g(size(b\_remove\_from\_free))) \mathbin{⩥} \{b\_remove\_from\_free\}) \cup$
$(\{-1\} \mathbin{⩤} \{nx(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free) \mapsto$
$pr(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free)\})$

end

Event *remove_from_free_2* $\widehat{=}$

refines *remove_from_free_2*

when

grd1 : $prog = call\_remove\_from\_free$

grd4 : $f(g(size(b\_remove\_from\_free))) = b\_remove\_from\_free$

then

act1 : $free\_bit(b\_remove\_from\_free) := FALSE$

act2 : $nx(g(size(b\_remove\_from\_free))) := \{b\_remove\_from\_free\} \mathbin{⩤} nx(g(size(b\_remove\_from\_free)))$

act3 : $f(g(size(b\_remove\_from\_free))) := (nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)$

act4 : $pr(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free,$
$(nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)\}$
$\mathbin{⩤} pr(g(size(b\_remove\_from\_free)))) \cup$
$(\{-1\} \mathbin{⩤} \{(nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free) \mapsto$
$(pr(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)\})$
act5 : $prog := return\_remove\_from\_free$

end

Event $reduce\_create \mathrel{\widehat{=}}$
extends $reduce\_create$

when

grd6 : $prog = call\_reduce\_create$

then

act1 : $size := (\{b\_reduce\_create\} \mathbin{⩤} size) \cup$
$\{b\_reduce\_create \mapsto q\_reduce\_create\} \cup$
$\{b\_reduce\_create + q\_reduce\_create \mapsto size(b\_reduce\_create) - q\_reduce\_create\}$
act3 : $left := (\{left^{-1}(b\_reduce\_create)\} \mathbin{⩤} left) \cup$
$\{b\_reduce\_create + q\_reduce\_create \mapsto b\_reduce\_create\} \cup$
$\{left^{-1}(b\_reduce\_create) \mapsto b\_reduce\_create + q\_reduce\_create\}$
act4 : $free\_bit(b\_reduce\_create + q\_reduce\_create) := FALSE$
act5 : $prog := return\_reduce\_create$

end

Event $merge\_right \mathrel{\widehat{=}}$
refines $merge\_right$

when

grd6 : $prog = call\_merge\_right$

then

act1 : $size := (\{left^{-1}(b\_merge\_right), b\_merge\_right\} \mathbin{⩤} size) \cup$
$\{b\_merge\_right \mapsto size(b\_merge\_right) + size(left^{-1}(b\_merge\_right))\}$
act3 : $left := (\{left^{-1}(b\_merge\_right)\} \mathbin{⩤} left \mathbin{⩥} \{left^{-1}(b\_merge\_right)\}) \cup$
$\{left^{-1}(left^{-1}(b\_merge\_right)) \mapsto b\_merge\_right\}$
act4 : $free\_bit := \{left^{-1}(b\_merge\_right)\} \mathbin{⩤} free\_bit$
act5 : $prog := return\_merge\_right$

end

END

2.10 m08: Introducing Search (One Dimensional Array)

An Event-B Specification of m08
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m08

Introducing search (one dimensional array)

REFINES m07

SEES c04

VARIABLES

$size$

$left$

nx

pr

f

$free\_bit$

$prog$

$b\_remove\_from\_free$

b_reduce_create

q_reduce_create

q_loc

b_make_free

$bloc$

b_merge_right

$search\_bit$

$bloc\_0$

q_loc_0

$adrp$

INVARIANTS

inv1 : $search\_bit \in BOOL$

inv2 : $bloc\_0 \in \mathbb{Z}$

inv3 : $q\_loc\_0 \in 0 \mathrel{..} m$

inv12 : $search\_bit = TRUE \land adr = 0 \Rightarrow bloc\_0 \in dom(size) \land free\_bit(bloc\_0) = TRUE \land q\_loc\_0 \in 1 \mathrel{..} m$

inv5 : $q\_loc\_0 = 0 \Rightarrow search\_bit = FALSE$

inv6 : $search\_bit = FALSE \Leftrightarrow bloc\_0 = -1$

inv7 : $adrp \in 6 \mathrel{..} 12 \Rightarrow search\_bit = FALSE$

inv8 : $adrp \in 0 \mathrel{..} 12$

inv9 : $adrp = 0 \Leftrightarrow search\_bit = FALSE \land adr = 0$

inv10 : $adrp = 1 \Leftrightarrow search\_bit = TRUE \land adr = 0$

inv11 : $adrp \in 2 \mathrel{..} 12 \Rightarrow adr = adrp - 1$

EVENTS

Initialisation

begin

act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$

act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$

act7 : $f := ((1 \mathrel{..} d-1) \times \{-1\}) \cup \{d \mapsto 1\}$

act8 : $nx := ((1 \mathrel{..} d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$

act9 : $pr := ((1 \mathrel{..} d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$

act10 : $free\_bit := \{0 \mapsto FALSE, 1 \mapsto TRUE, m+1 \mapsto FALSE\}$

act11 : $prog := undefined$

act13 : $b\_remove\_from\_free := 0$

act14 : $b\_reduce\_create := 0$

act15 : $q\_reduce\_create := 0$
act16 : $q\_loc := 0$
act17 : $b\_make\_free := 0$
act18 : $bloc := 0$
act19 : $b\_merge\_right := 0$
act20 : $search\_bit := FALSE$
act21 : $bloc\_0 := -1$
act22 : $q\_loc\_0 := 0$
act23 : $adrp := 0$

end

Event *search_fail* $\widehat{=}$

any

$q0$

where

grd12 : $q0 \in 1 \,..\, lower(d)$
grd1 : $\{i|i \in g\_srh(q0) \,..\, d \wedge f(i) \neq -1\} = \varnothing$
grd13 : $q\_loc\_0 = 0$
grd14 : $adrp = 0$

then

act1 : $q\_loc\_0 := 0$
act2 : $bloc\_0 := -1$

end

Event *search_success* $\widehat{=}$

any

j
$q0$

where

grd12 : $q0 \in 1 \,..\, lower(d)$
grd5 : $j \in 1 \,..\, d$
grd6 : $\{i|i \in g\_srh(q0) \,..\, d \wedge f(i) \neq -1\} \neq \varnothing$
grd1 : $j = min(\{i|i \in g\_srh(q0) \,..\, d \wedge f(i) \neq -1\})$
grd2 : $f(j) \neq -1$
grd7 : $q0 \leq lower(j)$
grd13 : $q\_loc\_0 = 0$
grd14 : $adrp = 0$

then

act1 : $bloc\_0 := f(j)$
act2 : $search\_bit := TRUE$
act3 : $q\_loc\_0 := q0$
act4 : $adrp := 1$

end

Event *allocate_1_1* $\widehat{=}$
refines *allocate_1_1*

when

grd9 : $adrp = 1$
grd8 : $bloc\_0 \neq -1$
grd4 : $q\_loc\_0 < size(bloc\_0)$
grd5 : $q\_loc\_0 > 0$

with

b : $b = bloc\_0$
q : $q = q\_loc\_0$

then

act2 : $prog := call\_remove\_from\_free$

act3 : $b\_remove\_from\_free := bloc\_0$
act4 : $q\_loc := q\_loc\_0$
act5 : $adrp := 3$

end

Event *allocate_1_2* $\widehat{=}$
refines *allocate_1_2*

when

grd1 : $prog = return\_remove\_from\_free$
grd3 : $adrp = 3$

then

act2 : $prog := call\_reduce\_create$
act3 : $b\_reduce\_create := b\_remove\_from\_free$
act4 : $q\_reduce\_create := q\_loc$
act5 : $adrp := 4$

end

Event *allocate_1_3* $\widehat{=}$
refines *allocate_1_3*

when

grd1 : $prog = return\_reduce\_create$
grd3 : $adrp = 4$

then

act2 : $prog := call\_make\_free$
act3 : $b\_make\_free := b\_reduce\_create + q\_loc$
act4 : $adrp := 5$

end

Event *allocate_1_4* $\widehat{=}$
refines *allocate_1_4*

when

grd1 : $prog = return\_make\_free$
grd3 : $adrp = 5$

then

act2 : $prog := undefined$
act3 : $search\_bit := FALSE$
act4 : $q\_loc\_0 := 0$
act5 : $bloc\_0 := -1$
act6 : $adrp := 0$

end

Event *allocate_2_1* $\widehat{=}$
refines *allocate_2_1*

when

grd6 : $bloc\_0 \neq -1$
grd7 : $adrp = 1$
grd3 : $q\_loc\_0 = size(bloc\_0)$

with

b : $b = bloc\_0$
q : $q = q\_loc\_0$

then

act2 : $prog := call\_remove\_from\_free$
act3 : $b\_remove\_from\_free := bloc\_0$
act4 : $adrp := 2$

end

Event *allocate_2_2* $\widehat{=}$
refines *allocate_2_2*

when

grd1 : $prog = return\_remove\_from\_free$
grd3 : $adrp = 2$

then

act1 : $prog := undefined$
act3 : $search\_bit := FALSE$
act4 : $q\_loc\_0 := 0$
act5 : $bloc\_0 := -1$
act6 : $adrp := 0$

end

Event *free_1_1* $\widehat{=}$
refines *free_1_1*

any

b

where

grd1 : $b \in dom(size)$
grd2 : $free\_bit(b) = FALSE$
grd3 : $b \notin \{0, m+1\}$
grd4 : $left(b) \in dom(size)$
grd5 : $free\_bit(left(b)) = TRUE$
grd8 : $adrp = 0$

then

act2 : $prog := call\_remove\_from\_free$
act3 : $b\_remove\_from\_free := left(b)$
act4 : $bloc := b$
act5 : $adrp := 6$

end

Event *free_1_2* $\widehat{=}$
refines *free_1_2*

when

grd1 : $prog = return\_remove\_from\_free$
grd3 : $adrp = 6$

then

act2 : $prog := call\_merge\_right$
act3 : $b\_merge\_right := left(bloc)$
act4 : $bloc := left(bloc)$
act5 : $adrp := 7$

end

Event *free_1_3* $\widehat{=}$
refines *free_1_3*

when

grd2 : $prog = return\_merge\_right$
grd3 : $adrp = 7$

then

act2 : $prog := undefined$
act3 : $adrp := 8$

end

Event *free_2* $\widehat{=}$
refines *free_2*

any

b

where

grd1 : $b \in dom(size)$

```
            grd2 : $free\_bit(b) = FALSE$
            grd3 : $b \notin \{0, m + 1\}$
            grd4 : $left(b) \in dom(size)$
            grd5 : $free\_bit(left(b)) = FALSE$
            grd8 : $adrp = 0$
      then
            act2 : $bloc := b$
            act3 : $adrp := 8$
      end
Event   free_3_1 ≙
refines free_3_1
      when
            grd4 : $adrp = 8$
            grd3 : $left^{-1}(bloc) = bloc + size(bloc)$
            grd2 : $free\_bit(bloc + size(bloc)) = TRUE$
      then
            act2 : $prog := call\_remove\_from\_free$
            act3 : $b\_remove\_from\_free := bloc + size(bloc)$
                   left$^{-1}$ (bloc)
            act4 : $adrp := 9$
      end
Event   free_3_2 ≙
refines free_3_2
      when
            grd2 : $prog = return\_remove\_from\_free$
            grd3 : $adrp = 9$
      then
            act2 : $prog := call\_merge\_right$
            act3 : $b\_merge\_right := bloc$
            act4 : $adrp := 10$
      end
Event   free_3_3 ≙
refines free_3_3
      when
            grd2 : $prog = return\_merge\_right$
            grd3 : $adrp = 10$
      then
            act2 : $prog := undefined$
            act3 : $adrp := 11$
      end
Event   free_4 ≙
refines free_4
      when
            grd5 : $adrp = 8$
            grd4 : $left^{-1}(bloc) = bloc + size(bloc)$
            grd2 : $bloc + size(bloc) \in dom(size)$
            grd3 : $free\_bit(bloc + size(bloc)) = FALSE$
                   left$^{-1}$ (bloc)
      then
            act2 : $adrp := 11$
      end
Event   free_5 ≙
refines free_5
```

when

grd2 : $adrp = 11$

then

act2 : $prog := call\_make\_free$
act3 : $b\_make\_free := bloc$
act4 : $adrp := 12$

end

Event *free_6* $\widehat{=}$
refines *free_6*

when

grd2 : $prog = return\_make\_free$
grd3 : $adrp = 12$

then

act2 : $prog := undefined$
act3 : $adrp := 0$

end

Event *make_free* $\widehat{=}$
extends *make_free*

when

grd6 : $prog = call\_make\_free$

then

act1 : $free\_bit(b\_make\_free) := TRUE$
act6 : $f(g(size(b\_make\_free))) := b\_make\_free$
act3 : $nx(g(size(b\_make\_free))) := nx(g(size(b\_make\_free))) \cup \{b\_make\_free \mapsto f(g(size(b\_make\_free)))\}$
act5 : $pr(g(size(b\_make\_free))) := (\{f(g(size(b\_make\_free)))\} ⩤ pr(g(size(b\_make\_free)))) \cup (\{-1\} ⩤ (\{f(g(size(b\_make\_free))) \mapsto b\_make\_free, b\_make\_free \mapsto -1\}))$
act4 : $prog := return\_make\_free$

end

Event *remove_from_free_1* $\widehat{=}$
extends *remove_from_free_1*

when

grd3 : $prog = call\_remove\_from\_free$
grd4 : $f(g(size(b\_remove\_from\_free))) \neq b\_remove\_from\_free$

then

act1 : $free\_bit(b\_remove\_from\_free) := FALSE$
act3 : $nx(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} ⩤ nx(g(size(b\_remove\_from\_free))) ⩥ \{b\_remove\_from\_free\}) \cup \{(pr(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free) \mapsto nx(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free)\}$
act4 : $prog := return\_remove\_from\_free$
act5 : $pr(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} ⩤ pr(g(size(b\_remove\_from\_free))) ⩥ \{b\_remove\_from\_free\}) \cup (\{-1\} ⩤ \{nx(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free) \mapsto pr(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free)\})$

end

Event *remove_from_free_2* $\widehat{=}$
extends *remove_from_free_2*

when

grd1 : $prog = call\_remove\_from\_free$
grd4 : $f(g(size(b\_remove\_from\_free))) = b\_remove\_from\_free$

then

act1 : $free\_bit(b\_remove\_from\_free) := FALSE$
act2 : $nx(g(size(b\_remove\_from\_free))) := \{b\_remove\_from\_free\} ⩤ nx(g(size(b\_remove\_from\_free)))$
act3 : $f(g(size(b\_remove\_from\_free))) := (nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)$

act4 : $\mathit{pr}(g(\mathit{size}(\mathit{b\_remove\_from\_free}))) := (\{\mathit{b\_remove\_from\_free}, (\mathit{nx}(g(\mathit{size}(\mathit{b\_remove\_from\_free}))))(\mathit{b\_remove\_from\_free})\} ⩤ \mathit{pr}(g(\mathit{size}(\mathit{b\_remove\_from\_free})))) \cup (\{-1\} ⩤ \{(\mathit{nx}(g(\mathit{size}(\mathit{b\_remove\_from\_free}))))(\mathit{b\_remove\_from\_free}) \mapsto (\mathit{pr}(g(\mathit{size}(\mathit{b\_remove\_from\_free}))))(\mathit{b\_remove\_from\_free})\})$

act5 : $\mathit{prog} := \mathit{return\_remove\_from\_free}$

end

Event *reduce_create* $\widehat{=}$

extends *reduce_create*

when

grd6 : $\mathit{prog} = \mathit{call\_reduce\_create}$

then

act1 : $\mathit{size} := (\{\mathit{b\_reduce\_create}\} ⩤ \mathit{size}) \cup \{\mathit{b\_reduce\_create} \mapsto \mathit{q\_reduce\_create}\} \cup \{\mathit{b\_reduce\_create} + \mathit{q\_reduce\_create} \mapsto \mathit{size}(\mathit{b\_reduce\_create}) - \mathit{q\_reduce\_create}\}$

act3 : $\mathit{left} := (\{\mathit{left}^{-1}(\mathit{b\_reduce\_create})\} ⩤ \mathit{left}) \cup \{\mathit{b\_reduce\_create} + \mathit{q\_reduce\_create} \mapsto \mathit{b\_reduce\_create}\} \cup \{\mathit{left}^{-1}(\mathit{b\_reduce\_create}) \mapsto \mathit{b\_reduce\_create} + \mathit{q\_reduce\_create}\}$

act4 : $\mathit{free\_bit}(\mathit{b\_reduce\_create} + \mathit{q\_reduce\_create}) := FALSE$

act5 : $\mathit{prog} := \mathit{return\_reduce\_create}$

end

Event *merge_right* $\widehat{=}$

extends *merge_right*

when

grd6 : $\mathit{prog} = \mathit{call\_merge\_right}$

then

act1 : $\mathit{size} := (\{\mathit{left}^{-1}(\mathit{b\_merge\_right}), \mathit{b\_merge\_right}\} ⩤ \mathit{size}) \cup \{\mathit{b\_merge\_right} \mapsto \mathit{size}(\mathit{b\_merge\_right}) + \mathit{size}(\mathit{left}^{-1}(\mathit{b\_merge\_right}))\}$

act3 : $\mathit{left} := (\{\mathit{left}^{-1}(\mathit{b\_merge\_right})\} ⩤ \mathit{left} ⩥ \{\mathit{left}^{-1}(\mathit{b\_merge\_right})\}) \cup \{\mathit{left}^{-1}(\mathit{left}^{-1}(\mathit{b\_merge\_right})) \mapsto \mathit{b\_merge\_right}\}$

act4 : $\mathit{free\_bit} := \{\mathit{left}^{-1}(\mathit{b\_merge\_right})\} ⩤ \mathit{free\_bit}$

act5 : $\mathit{prog} := \mathit{return\_merge\_right}$

end

END

2.11 m09: Simplifying Search (Still One Dimensional Array)

An Event-B Specification of m09
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m09
Simplifying search (still one dimensional array)
REFINES m08
SEES c05
VARIABLES

$size$
$left$
nx
pr
f
$free\_bit$
$prog$
$b\_remove\_from\_free$
b_reduce_create
q_reduce_create
q_loc
b_make_free
$bloc$
b_merge_right
$bloc\_0$
q_loc_0
$adrp$

INVARIANTS

inv1 : $adrp = 0 \lor adrp \in 6\mathrel{..}12 \Rightarrow q\_loc\_0 = 0$
inv2 : $bloc\_0 \neq -1 \Rightarrow q\_loc\_0 > 0$

EVENTS
Initialisation
begin

act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$
act7 : $f := ((1\mathrel{..}d-1) \times \{-1\}) \cup \{d \mapsto 1\}$
act8 : $nx := ((1\mathrel{..}d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act9 : $pr := ((1\mathrel{..}d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act10 : $free\_bit := \{0 \mapsto FALSE, 1 \mapsto TRUE, m+1 \mapsto FALSE\}$
act11 : $prog := undefined$
act13 : $b\_remove\_from\_free := 0$
act14 : $b\_reduce\_create := 0$
act15 : $q\_reduce\_create := 0$
act16 : $q\_loc := 0$
act17 : $b\_make\_free := 0$
act18 : $bloc := 0$
act19 : $b\_merge\_right := 0$
act21 : $bloc\_0 := -1$
act22 : $q\_loc\_0 := 0$
act23 : $adrp := 0$

end

Event $search\_fail \mathrel{\widehat{=}}$
refines $search\_fail$

any

$q0$

q

where

grd12 : $q0 \in 1 \,..\, lower(d)$

grd15 : $q = search\_start(q0)$

grd16 : $q \in 1 \,..\, m$

grd1 : $\{i | i \in g(q) \,..\, d \wedge f(i) \neq -1\} = \varnothing$

grd14 : $adrp = 0$

grd13 : $q\_loc\_0 = 0$

then

act2 : $bloc\_0 := -1$

end

Event *search_success* $\widehat{=}$

refines *search_success*

any

j

$q0$

q

where

grd12 : $q0 \in 1 \,..\, lower(d)$

grd15 : $q = search\_start(q0)$

grd16 : $q \in 1 \,..\, m$

grd5 : $j \in 1 \,..\, d$

grd6 : $\{i | i \in g(q) \,..\, d \wedge f(i) \neq -1\} \neq \varnothing$

grd7 : $j = min(\{i | i \in g(q) \,..\, d \wedge f(i) \neq -1\})$

grd14 : $adrp = 0$

grd13 : $q\_loc\_0 = 0$

then

act1 : $bloc\_0 := f(j)$

act3 : $q\_loc\_0 := q0$

act4 : $adrp := 1$

end

Event *allocate_1_1* $\widehat{=}$

refines *allocate_1_1*

when

grd9 : $adrp = 1$

grd8 : $bloc\_0 \neq -1$

grd4 : $q\_loc\_0 < size(bloc\_0)$

grd5 : $q\_loc\_0 > 0$

then

act2 : $prog := call\_remove\_from\_free$

act3 : $b\_remove\_from\_free := bloc\_0$

act4 : $q\_loc := q\_loc\_0$

act5 : $adrp := 3$

end

Event *allocate_1_2* $\widehat{=}$

extends *allocate_1_2*

when

grd1 : $prog = return\_remove\_from\_free$

grd3 : $adrp = 3$

then

act2 : $prog := call\_reduce\_create$

act3 : $b\_reduce\_create := b\_remove\_from\_free$

act4 : $q\_reduce\_create := q\_loc$
act5 : $adrp := 4$
end

Event *allocate_1_3* $\widehat{=}$
extends *allocate_1_3*
when
grd1 : $prog = return\_reduce\_create$
grd3 : $adrp = 4$
then
act2 : $prog := call\_make\_free$
act3 : $b\_make\_free := b\_reduce\_create + q\_loc$
act4 : $adrp := 5$
end

Event *allocate_1_4* $\widehat{=}$
refines *allocate_1_4*
when
grd1 : $prog = return\_make\_free$
grd3 : $adrp = 5$
then
act2 : $prog := undefined$
act4 : $q\_loc\_0 := 0$
act5 : $bloc\_0 := -1$
act6 : $adrp := 0$
end

Event *allocate_2_1* $\widehat{=}$
refines *allocate_2_1*
when
grd6 : $bloc\_0 \neq -1$
grd7 : $adrp = 1$
grd3 : $q\_loc\_0 = size(bloc\_0)$
then
act2 : $prog := call\_remove\_from\_free$
act3 : $b\_remove\_from\_free := bloc\_0$
act4 : $adrp := 2$
end

Event *allocate_2_2* $\widehat{=}$
refines *allocate_2_2*
when
grd1 : $prog = return\_remove\_from\_free$
grd3 : $adrp = 2$
then
act1 : $prog := undefined$
act4 : $q\_loc\_0 := 0$
act5 : $bloc\_0 := -1$
act6 : $adrp := 0$
end

Event *free_1_1* $\widehat{=}$
extends *free_1_1*
any
b
where
grd1 : $b \in dom(size)$

grd2 : $free\_bit(b) = FALSE$

grd3 : $b \notin \{0, m+1\}$

grd4 : $left(b) \in dom(size)$

grd5 : $free\_bit(left(b)) = TRUE$

grd8 : $adrp = 0$

then

act2 : $prog := call\_remove\_from\_free$

act3 : $b\_remove\_from\_free := left(b)$

act4 : $bloc := b$

act5 : $adrp := 6$

end

Event *free_1_2* $\widehat{=}$

extends *free_1_2*

when

grd1 : $prog = return\_remove\_from\_free$

grd3 : $adrp = 6$

then

act2 : $prog := call\_merge\_right$

act3 : $b\_merge\_right := left(bloc)$

act4 : $bloc := left(bloc)$

act5 : $adrp := 7$

end

Event *free_1_3* $\widehat{=}$

extends *free_1_3*

when

grd2 : $prog = return\_merge\_right$

grd3 : $adrp = 7$

then

act2 : $prog := undefined$

act3 : $adrp := 8$

end

Event *free_2* $\widehat{=}$

extends *free_2*

any

b

where

grd1 : $b \in dom(size)$

grd2 : $free\_bit(b) = FALSE$

grd3 : $b \notin \{0, m+1\}$

grd4 : $left(b) \in dom(size)$

grd5 : $free\_bit(left(b)) = FALSE$

grd8 : $adrp = 0$

then

act2 : $bloc := b$

act3 : $adrp := 8$

end

Event *free_3_1* $\widehat{=}$

extends *free_3_1*

when

grd4 : $adrp = 8$

grd3 : $left^{-1}(bloc) = bloc + size(bloc)$

grd2 : $free\_bit(bloc + size(bloc)) = TRUE$

then

act2 : $prog := call\_remove\_from\_free$
act3 : $b\_remove\_from\_free := bloc + size(bloc)$
left^{-1} (bloc)
act4 : $adrp := 9$
end

Event *free_3_2* $\widehat{=}$
extends *free_3_2*
when
grd2 : $prog = return\_remove\_from\_free$
grd3 : $adrp = 9$
then
act2 : $prog := call\_merge\_right$
act3 : $b\_merge\_right := bloc$
act4 : $adrp := 10$
end

Event *free_3_3* $\widehat{=}$
extends *free_3_3*
when
grd2 : $prog = return\_merge\_right$
grd3 : $adrp = 10$
then
act2 : $prog := undefined$
act3 : $adrp := 11$
end

Event *free_4* $\widehat{=}$
extends *free_4*
when
grd5 : $adrp = 8$
grd4 : $left^{-1}(bloc) = bloc + size(bloc)$
grd2 : $bloc + size(bloc) \in dom(size)$
grd3 : $free\_bit(bloc + size(bloc)) = FALSE$
left^{-1} (bloc)
then
act2 : $adrp := 11$
end

Event *free_5* $\widehat{=}$
extends *free_5*
when
grd2 : $adrp = 11$
then
act2 : $prog := call\_make\_free$
act3 : $b\_make\_free := bloc$
act4 : $adrp := 12$
end

Event *free_6* $\widehat{=}$
extends *free_6*
when
grd2 : $prog = return\_make\_free$
grd3 : $adrp = 12$
then
act2 : $prog := undefined$
act3 : $adrp := 0$

end

Event *make_free* $\widehat{=}$
extends *make_free*

when

grd6 : $prog = call\_make\_free$

then

act1 : $free\_bit(b\_make\_free) := TRUE$
act6 : $f(g(size(b\_make\_free))) := b\_make\_free$
act3 : $nx(g(size(b\_make\_free))) := nx(g(size(b\_make\_free))) \cup \{b\_make\_free \mapsto f(g(size(b\_make\_free)))\}$
act5 : $pr(g(size(b\_make\_free))) := (\{f(g(size(b\_make\_free)))\} ⩤ pr(g(size(b\_make\_free)))) \cup (\{-1\} ⩤ (\{f(g(size(b\_make\_free))) \mapsto b\_make\_free, b\_make\_free \mapsto -1\}))$
act4 : $prog := return\_make\_free$

end

Event *remove_from_free_1* $\widehat{=}$
extends *remove_from_free_1*

when

grd3 : $prog = call\_remove\_from\_free$
grd4 : $f(g(size(b\_remove\_from\_free))) \neq b\_remove\_from\_free$

then

act1 : $free\_bit(b\_remove\_from\_free) := FALSE$
act3 : $nx(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} ⩤ nx(g(size(b\_remove\_from\_free))) ⩥ \{b\_remove\_from\_free\}) \cup \{(pr(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free) \mapsto nx(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free)\}$
act4 : $prog := return\_remove\_from\_free$
act5 : $pr(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} ⩤ pr(g(size(b\_remove\_from\_free))) ⩥ \{b\_remove\_from\_free\}) \cup (\{-1\} ⩤ \{nx(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free) \mapsto pr(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free)\})$

end

Event *remove_from_free_2* $\widehat{=}$
extends *remove_from_free_2*

when

grd1 : $prog = call\_remove\_from\_free$
grd4 : $f(g(size(b\_remove\_from\_free))) = b\_remove\_from\_free$

then

act1 : $free\_bit(b\_remove\_from\_free) := FALSE$
act2 : $nx(g(size(b\_remove\_from\_free))) := \{b\_remove\_from\_free\} ⩤ nx(g(size(b\_remove\_from\_free)))$
act3 : $f(g(size(b\_remove\_from\_free))) := (nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)$
act4 : $pr(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free, (nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)\} ⩤ pr(g(size(b\_remove\_from\_free)))) \cup (\{-1\} ⩤ \{(nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free) \mapsto (pr(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)\})$
act5 : $prog := return\_remove\_from\_free$

end

Event *reduce_create* $\widehat{=}$
extends *reduce_create*

when

grd6 : $prog = call\_reduce\_create$

then

act1 : $size := (\{b\_reduce\_create\} ⩤ size) \cup \{b\_reduce\_create \mapsto q\_reduce\_create\} \cup \{b\_reduce\_create + q\_reduce\_create \mapsto size(b\_reduce\_create) - q\_reduce\_create\}$
act3 : $left := (\{left^{-1}(b\_reduce\_create)\} ⩤ left) \cup \{b\_reduce\_create + q\_reduce\_create \mapsto b\_reduce\_create\} \cup \{left^{-1}(b\_reduce\_create) \mapsto b\_reduce\_create + q\_reduce\_create\}$
act4 : $free\_bit(b\_reduce\_create + q\_reduce\_create) := FALSE$
act5 : $prog := return\_reduce\_create$

end

Event $merge\_right \mathrel{\widehat{=}}$

extends $merge\_right$

when

grd6 : $prog = call\_merge\_right$

then

act1 : $size := (\{left^{-1}(b\_merge\_right), b\_merge\_right\} ⩤ size) \cup \{b\_merge\_right \mapsto size(b\_merge\_right) + size(left^{-1}(b\_merge\_right))\}$

act3 : $left := (\{left^{-1}(b\_merge\_right)\} ⩤ left ⩥ \{left^{-1}(b\_merge\_right)\}) \cup \{left^{-1}(left^{-1}(b\_merge\_right)) \mapsto b\_merge\_right\}$

act4 : $free\_bit := \{left^{-1}(b\_merge\_right)\} ⩤ free\_bit$

act5 : $prog := return\_merge\_right$

end

END

2.12 m10: Introducing Two Dimensional Array for Search

An Event-B Specification of m10
Creation Date: 4Jul2015 @ 10:10:16 PM

MACHINE m10

Introducing two dimensional array for search.

REFINES m09
SEES c05
VARIABLES

$size$
$left$
nx
pr
f
$free\_bit$
$prog$
$b\_remove\_from\_free$
b_reduce_create
q_reduce_create
q_loc
b_make_free
$bloc$
b_merge_right
$bloc\_0$
q_loc_0
$adrp$

EVENTS
Initialisation

extended

begin

act2 : $size := \{0 \mapsto 1, 1 \mapsto m, m+1 \mapsto 1\}$
act4 : $left := \{1 \mapsto 0, m+1 \mapsto 1\}$
act7 : $f := ((1 \,..\, d-1) \times \{-1\}) \cup \{d \mapsto 1\}$
act8 : $nx := ((1 \,..\, d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act9 : $pr := ((1 \,..\, d-1) \times \{\varnothing\}) \cup \{d \mapsto \{1 \mapsto -1\}\}$
act10 : $free\_bit := \{0 \mapsto FALSE, 1 \mapsto TRUE, m+1 \mapsto FALSE\}$
act11 : $prog := undefined$
act13 : $b\_remove\_from\_free := 0$
act14 : $b\_reduce\_create := 0$
act15 : $q\_reduce\_create := 0$
act16 : $q\_loc := 0$
act17 : $b\_make\_free := 0$
act18 : $bloc := 0$
act19 : $b\_merge\_right := 0$
act21 : $bloc\_0 := -1$
act22 : $q\_loc\_0 := 0$
act23 : $adrp := 0$

end

Event $search\_fail \mathrel{\widehat{=}}$
refines $search\_fail$

any

$q0$
q

where

grd12 : $q0 \in 1 \mathinner{..} lower(d)$

grd15 : $q = search\_start(q0)$

grd16 : $q \in 1 \mathinner{..} m$

grd18 : $\{i | i \in g(q) \mathinner{..} g(fl(q) * ms + ms - 1) \land f(i) \neq -1\} = \varnothing$

grd19 : $\{r | r \in fl(q) + 1 \mathinner{..} mf - 1 \land \{i | i \in g(r * ms) \mathinner{..} g(r * ms + ms - 1) \land f(i) \neq -1\} \neq \varnothing\} = \varnothing$

grd1 : $\{i | i \in g(q) \mathinner{..} d \land f(i) \neq -1\} = \varnothing$

grd14 : $adrp = 0$

grd13 : $q\_loc\_0 = 0$

then

act2 : $bloc\_0 := -1$

end

Event *search_success_1* $\widehat{=}$

refines *search_success*

any

j

$q0$

q

where

grd12 : $q0 \in 1 \mathinner{..} lower(d)$

grd15 : $q = search\_start(q0)$

grd16 : $q \in 1 \mathinner{..} m$

grd11 : $q = fl(q) * ms + ft(q)$

grd19 : $\{i | i \in g(q) \mathinner{..} g(fl(q) * ms + ms - 1) \land f(i) \neq -1\} \neq \varnothing$

grd18 : $g(fl(q) * ms + ms - 1) \leq d$

grd5 : $j \in 1 \mathinner{..} d$

grd6 : $\{i | i \in g(q) \mathinner{..} d \land f(i) \neq -1\} \neq \varnothing$

grd7 : $j = min(\{i | i \in g(q) \mathinner{..} d \land f(i) \neq -1\})$

grd17 : $j = min(\{i | i \in g(q) \mathinner{..} g(fl(q) * ms + ms - 1) \land f(i) \neq -1\})$

grd14 : $adrp = 0$

then

act1 : $bloc\_0 := f(j)$

act3 : $q\_loc\_0 := q0$

act4 : $adrp := 1$

end

Event *search_success_2* $\widehat{=}$

refines *search_success*

any

j

$q0$

k

q

where

grd12 : $q0 \in 1 \mathinner{..} lower(d)$

grd15 : $q = search\_start(q0)$

grd16 : $q \in 1 \mathinner{..} m$

grd11 : $q = fl(q) * ms + ft(q)$

grd10 : $\{i | i \in g(q) \mathinner{..} g(fl(q) * ms + ms - 1) \land f(i) \neq -1\} = \varnothing$

grd17 : $g(fl(q) * ms + ms - 1) \leq d$

grd5 : $j \in 1 \mathinner{..} d$

grd6 : $\{i | i \in g(q) \mathinner{..} d \land f(i) \neq -1\} \neq \varnothing$

grd7 : $j = min(\{i | i \in g(q) \mathinner{..} d \land f(i) \neq -1\})$

grd18 : $\{r | r \in fl(q) + 1 \mathinner{..} mf - 1 \land \{i | i \in g(r * ms) \mathinner{..} g(r * ms + ms - 1) \land f(i) \neq -1\} \neq \varnothing\} \neq \varnothing$

grd19 : $k = min(\{r | r \in fl(q) + 1 \mathinner{..} mf - 1 \land \{i | i \in g(r * ms) \mathinner{..} g(r * ms + ms - 1) \land f(i) \neq -1\} \neq \varnothing\})$

grd20 : $\{i | i \in g(k * ms) \mathinner{..} g(k * ms + ms - 1) \land f(i) \neq -1\} \neq \varnothing$

grd21 : $j = min(\{i|i \in g(k*ms) \mathinner{..} g(k*ms+ms-1) \wedge f(i) \neq -1\})$
grd14 : $adrp = 0$
grd13 : $q\_loc\_0 = 0$

then

act1 : $bloc\_0 := f(j)$
act3 : $q\_loc\_0 := q0$
act4 : $adrp := 1$

end

Event *allocate_1_1* $\widehat{=}$
extends *allocate_1_1*

when

grd9 : $adrp = 1$
grd8 : $bloc\_0 \neq -1$
grd4 : $q\_loc\_0 < size(bloc\_0)$
grd5 : $q\_loc\_0 > 0$

then

act2 : $prog := call\_remove\_from\_free$
act3 : $b\_remove\_from\_free := bloc\_0$
act4 : $q\_loc := q\_loc\_0$
act5 : $adrp := 3$

end

Event *allocate_1_2* $\widehat{=}$
extends *allocate_1_2*

when

grd1 : $prog = return\_remove\_from\_free$
grd3 : $adrp = 3$

then

act2 : $prog := call\_reduce\_create$
act3 : $b\_reduce\_create := b\_remove\_from\_free$
act4 : $q\_reduce\_create := q\_loc$
act5 : $adrp := 4$

end

Event *allocate_1_3* $\widehat{=}$
extends *allocate_1_3*

when

grd1 : $prog = return\_reduce\_create$
grd3 : $adrp = 4$

then

act2 : $prog := call\_make\_free$
act3 : $b\_make\_free := b\_reduce\_create + q\_loc$
act4 : $adrp := 5$

end

Event *allocate_1_4* $\widehat{=}$
extends *allocate_1_4*

when

grd1 : $prog = return\_make\_free$
grd3 : $adrp = 5$

then

act2 : $prog := undefined$
act4 : $q\_loc\_0 := 0$
act5 : $bloc\_0 := -1$
act6 : $adrp := 0$

end

Event *allocate_2_1* $\widehat{=}$
extends *allocate_2_1*

when

grd6 : $bloc\_0 \neq -1$
grd7 : $adrp = 1$
grd3 : $q\_loc\_0 = size(bloc\_0)$

then

act2 : $prog := call\_remove\_from\_free$
act3 : $b\_remove\_from\_free := bloc\_0$
act4 : $adrp := 2$

end

Event *allocate_2_2* $\widehat{=}$
extends *allocate_2_2*

when

grd1 : $prog = return\_remove\_from\_free$
grd3 : $adrp = 2$

then

act1 : $prog := undefined$
act4 : $q\_loc\_0 := 0$
act5 : $bloc\_0 := -1$
act6 : $adrp := 0$

end

Event *free_1_1* $\widehat{=}$
extends *free_1_1*

any

b

where

grd1 : $b \in dom(size)$
grd2 : $free\_bit(b) = FALSE$
grd3 : $b \notin \{0, m + 1\}$
grd4 : $left(b) \in dom(size)$
grd5 : $free\_bit(left(b)) = TRUE$
grd8 : $adrp = 0$

then

act2 : $prog := call\_remove\_from\_free$
act3 : $b\_remove\_from\_free := left(b)$
act4 : $bloc := b$
act5 : $adrp := 6$

end

Event *free_1_2* $\widehat{=}$
extends *free_1_2*

when

grd1 : $prog = return\_remove\_from\_free$
grd3 : $adrp = 6$

then

act2 : $prog := call\_merge\_right$
act3 : $b\_merge\_right := left(bloc)$
act4 : $bloc := left(bloc)$
act5 : $adrp := 7$

end

Event *free_1_3* $\widehat{=}$
extends *free_1_3*

when

grd2 : $prog = return\_merge\_right$
grd3 : $adrp = 7$

then

act2 : $prog := undefined$
act3 : $adrp := 8$

end

Event *free_2* $\widehat{=}$
extends *free_2*

any

b

where

grd1 : $b \in dom(size)$
grd2 : $free\_bit(b) = FALSE$
grd3 : $b \notin \{0, m + 1\}$
grd4 : $left(b) \in dom(size)$
grd5 : $free\_bit(left(b)) = FALSE$
grd8 : $adrp = 0$

then

act2 : $bloc := b$
act3 : $adrp := 8$

end

Event *free_3_1* $\widehat{=}$
extends *free_3_1*

when

grd4 : $adrp = 8$
grd3 : $left^{-1}(bloc) = bloc + size(bloc)$
grd2 : $free\_bit(bloc + size(bloc)) = TRUE$

then

act2 : $prog := call\_remove\_from\_free$
act3 : $b\_remove\_from\_free := bloc + size(bloc)$
left^{-1} (bloc)
act4 : $adrp := 9$

end

Event *free_3_2* $\widehat{=}$
extends *free_3_2*

when

grd2 : $prog = return\_remove\_from\_free$
grd3 : $adrp = 9$

then

act2 : $prog := call\_merge\_right$
act3 : $b\_merge\_right := bloc$
act4 : $adrp := 10$

end

Event *free_3_3* $\widehat{=}$
extends *free_3_3*

when

grd2 : $prog = return\_merge\_right$
grd3 : $adrp = 10$

then

act2 : $prog := undefined$
act3 : $adrp := 11$

end

Event *free_4* $\widehat{=}$

extends *free_4*

when

grd5 : $adrp = 8$

grd4 : $left^{-1}(bloc) = bloc + size(bloc)$

grd2 : $bloc + size(bloc) \in dom(size)$

grd3 : $free\_bit(bloc + size(bloc)) = FALSE$

left^{-1} (bloc)

then

act2 : $adrp := 11$

end

Event *free_5* $\widehat{=}$

extends *free_5*

when

grd2 : $adrp = 11$

then

act2 : $prog := call\_make\_free$

act3 : $b\_make\_free := bloc$

act4 : $adrp := 12$

end

Event *free_6* $\widehat{=}$

extends *free_6*

when

grd2 : $prog = return\_make\_free$

grd3 : $adrp = 12$

then

act2 : $prog := undefined$

act3 : $adrp := 0$

end

Event *make_free* $\widehat{=}$

extends *make_free*

when

grd6 : $prog = call\_make\_free$

then

act1 : $free\_bit(b\_make\_free) := TRUE$

act6 : $f(g(size(b\_make\_free))) := b\_make\_free$

act3 : $nx(g(size(b\_make\_free))) := nx(g(size(b\_make\_free))) \cup \{b\_make\_free \mapsto f(g(size(b\_make\_free)))\}$

act5 : $pr(g(size(b\_make\_free))) := (\{f(g(size(b\_make\_free)))\} ⩤ pr(g(size(b\_make\_free)))) \cup (\{-1\} ⩤$
$(\{f(g(size(b\_make\_free))) \mapsto b\_make\_free, b\_make\_free \mapsto -1\}))$

act4 : $prog := return\_make\_free$

end

Event *remove_from_free_1* $\widehat{=}$

extends *remove_from_free_1*

when

grd3 : $prog = call\_remove\_from\_free$

grd4 : $f(g(size(b\_remove\_from\_free))) \neq b\_remove\_from\_free$

then

act1 : $free\_bit(b\_remove\_from\_free) := FALSE$

act3 : $nx(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} ⩤ nx(g(size(b\_remove\_from\_free))) ⩥$
$\{b\_remove\_from\_free\}) \cup \{(pr(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free) \mapsto$
$nx(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free)\}$

act4 : $prog := return\_remove\_from\_free$

act5 : $pr(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free\} ⩤ pr(g(size(b\_remove\_from\_free))) ⩥ \{b\_remove\_from\_free\}) \cup (\{-1\} ⩤ \{nx(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free) \mapsto pr(g(size(b\_remove\_from\_free)))(b\_remove\_from\_free)\})$

end

Event *remove_from_free_2* $\widehat{=}$
extends *remove_from_free_2*

when

grd1 : $prog = call\_remove\_from\_free$
grd4 : $f(g(size(b\_remove\_from\_free))) = b\_remove\_from\_free$

then

act1 : $free\_bit(b\_remove\_from\_free) := FALSE$
act2 : $nx(g(size(b\_remove\_from\_free))) := \{b\_remove\_from\_free\} ⩤ nx(g(size(b\_remove\_from\_free)))$
act3 : $f(g(size(b\_remove\_from\_free))) := (nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)$
act4 : $pr(g(size(b\_remove\_from\_free))) := (\{b\_remove\_from\_free, (nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)\} ⩤ pr(g(size(b\_remove\_from\_free)))) \cup (\{-1\} ⩤ \{(nx(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free) \mapsto (pr(g(size(b\_remove\_from\_free))))(b\_remove\_from\_free)\})$
act5 : $prog := return\_remove\_from\_free$

end

Event *reduce_create* $\widehat{=}$
extends *reduce_create*

when

grd6 : $prog = call\_reduce\_create$

then

act1 : $size := (\{b\_reduce\_create\} ⩤ size) \cup \{b\_reduce\_create \mapsto q\_reduce\_create\} \cup \{b\_reduce\_create + q\_reduce\_create \mapsto size(b\_reduce\_create) - q\_reduce\_create\}$
act3 : $left := (\{left^{-1}(b\_reduce\_create)\} ⩤ left) \cup \{b\_reduce\_create + q\_reduce\_create \mapsto b\_reduce\_create\} \cup \{left^{-1}(b\_reduce\_create) \mapsto b\_reduce\_create + q\_reduce\_create\}$
act4 : $free\_bit(b\_reduce\_create + q\_reduce\_create) := FALSE$
act5 : $prog := return\_reduce\_create$

end

Event *merge_right* $\widehat{=}$
extends *merge_right*

when

grd6 : $prog = call\_merge\_right$

then

act1 : $size := (\{left^{-1}(b\_merge\_right), b\_merge\_right\} ⩤ size) \cup \{b\_merge\_right \mapsto size(b\_merge\_right) + size(left^{-1}(b\_merge\_right))\}$
act3 : $left := (\{left^{-1}(b\_merge\_right)\} ⩤ left ⩥ \{left^{-1}(b\_merge\_right)\}) \cup \{left^{-1}(left^{-1}(b\_merge\_right)) \mapsto b\_merge\_right\}$
act4 : $free\_bit := \{left^{-1}(b\_merge\_right)\} ⩤ free\_bit$
act5 : $prog := return\_merge\_right$

end

END