1 Heap level 0 version 0

section *Heap*0 **parents** *standard_toolkit*

$Loc == \mathbb{N}$
$Free0 == \mathbb{P}\ Loc$

relation(**is_sequential** _)
relation(**has_seq** _)

generic(**partitionVDM** _)

$$\mathbf{partitionVDM}\ X == \{p : \mathbb{P}(\mathbb{P}\ X) \mid \bigcup p = X \land \{\} \notin p\}$$

$$\mathbf{is\_sequential}\ \_ == \{s : \mathrm{seq}\ \mathbb{N} \mid \exists i, j : Loc \bullet \mathrm{ran}\ s = i \mathinner{..} j\}$$
$$\mathbf{has\_seq}\ \_ == \{s : \mathrm{seq}\ Loc;\ n : \mathbb{N};\ f : Free0 \mid \mathbf{is\_sequential}\ s \land \mathrm{ran}\ s \subseteq f \land \mathrm{dom}\ s = 1 \mathinner{..} n\}$$

Schema *Heap*0

Z_Abs_St
$free0 : Free0$

$true$

Schema *NEW*0

$\Delta Heap0$
$req? : \mathbb{N}$
$res! : \mathbb{P}\ Loc$

$\exists s : \mathrm{seq}\ Loc \bullet$
$\quad \mathbf{has\_seq}\ (s, req?, free0) \land$
$\quad res! = \mathrm{ran}\ s \land$
$\quad free0' = free0 \setminus res!$

$DISPOSE0$

$$\Delta Heap0$$
$$ret? : Free0$$

$$ret? \cap free0 = \varnothing$$
$$free0' = free0 \cup ret?$$

2 VCs

$Heap0FSBSig$

$$Heap0$$

$$Heap0$$

$NEW0FSBSig$

$$Heap0$$
$$req? : \mathbb{N}$$

$$\exists\, t : \mathrm{seq}\ Loc \bullet \mathbf{has\_seq}\,(t, req?, free0)$$

$DISPOSE0FSBSig$

$$Heap0$$
$$ret? : Free0$$

$$ret? \cap free0 = \varnothing$$

theorem Heap0_ vc_ fsb_ state
$\exists\, Heap0FSBSig \mid true \bullet true$

theorem NEW0_ vc_ fsb_ pre
$\forall\, NEW0FSBSig \mid true \bullet \mathbf{pre}\, NEW0$

theorem DISPOSE0_ vc_ fsb_ pre
$\forall\, DISPOSE0FSBSig \mid true \bullet \mathbf{pre}\, DISPOSE0$

3 Proofs

proof[$Heap0\_\ vc\_\ fsb\_\ state$]
$\quad instantiate\ free0\ ==\ \varnothing;$
$\quad prove\ by\ reduce;$
$\quad \blacksquare$

proof[$NEW0\_\ vc\_\ fsb\_\ pre$]
$\qquad prove\ by\ reduce;$
$\qquad instantiate\ s\ ==\ t;$
$\qquad prove;$
$\qquad instantiate\ i\_\_0\ ==\ i,\ j\_\_0\ ==\ j;$
$\qquad prove;$
$\quad \blacksquare$

proof[$DISPOSE0\_\ vc\_\ fsb\_\ pre$]
$\quad prove\ by\ reduce;$
$\quad \blacksquare$

4 Heap level 1 version 0

Loc from *Heap*0 will be *Loc* in this name space too.

section *Heap*1 **parents** *Heap*0

$$Piece \mathrel{\widehat=} [\, LOC : Loc;\ SIZE : \mathbb{N}\,]$$

relation(**inv** − **Free** _)

$$locs\_of == (\lambda\, p : Piece \bullet p.LOC \mathrel{..} p.LOC + p.SIZE - 1)$$

theorem grule gLocsOfPFunType
$\quad locs\_of \in \langle\!| LOC : \mathbb{Z};\ SIZE : \mathbb{Z} |\!\rangle \pfun \mathbb{P}\,\mathbb{Z}$

theorem rule lLocsOfIsTotal
$\quad \forall\, p : Piece \bullet p \in \operatorname{dom} locs\_of$

$$
\begin{aligned}
&\mathbf{inv-Free}\ \_ == \{fr : \mathbb{P}\ Piece \mid \forall p1, p2 : fr \bullet \\
&\quad p1 = p2 \lor \\
&\quad (locs\_of\ (p1) \cap locs\_of\ (p2) = \varnothing \land \\
&\quad p1.LOC + p1.SIZE \neq p2.LOC)\} \\
&Free1 == \{ps : \mathbb{P}\ Piece \mid ps \in \mathbb{F}\ Piece \land \\
&\mathbf{inv-Free}\,(ps)\}
\end{aligned}
$$

Following Cliff's suggestion, will weaken *locs* to be on set of Piece rather than Free1

$$locs == (\lambda f : \mathbb{P}\ Piece \bullet \bigcup \{p : Piece \mid p \in f \bullet locs\_of\ (p)\})$$

theorem grule gLocsPFunType

$$locs \in \mathbb{P}\,(\lblot LOC : \mathbb{Z};\ SIZE : \mathbb{Z} \rblot) \pfun \mathbb{P}\,\mathbb{Z}$$

theorem rule lLocsIsTotal

$$\forall f : \mathbb{P}\ Piece \bullet f \in \mathrm{dom}\ locs$$

$$
\begin{array}{|l}
\hline
\textit{Heap1} \\
\hline
free1 : Free1 \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{NEW1} \\
\hline
\Delta Heap1 \\
req? : \mathbb{N} \\
res! : Piece \\
\hline
locs\,(free1') = locs\,(free1) \setminus locs\_of\ (res!) \\
locs\_of\ (res!) \subseteq locs\,(free1) \\
res!.SIZE = req? \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{DISPOSE1} \\
\hline
\Delta Heap1 \\
ret? : Piece \\
\hline
locs\_of\ (ret?) \cap locs\,(free1) = \varnothing \\
locs\,(free1') = locs\,(free1) \cup locs\_of(ret?) \\
\hline
\end{array}
$$

5 Lemmas

```
A10:?maybe not needed.
begin[disabled]{theorem}{rule dlInFree1ElemType}
fr \in Free1 \land p \in fr \implies  p \in Piece
end{theorem}
```

A11+A1

theorem disabled rule dlInFree1ElemType
$\forall fr : \mathbb{P}\ Piece \bullet \forall p : fr \bullet p \in Piece$

theorem frule fFree1ElemType
$f \in Free1 \Rightarrow f \in \mathbb{P}\ Piece$

A11

theorem grule gLocType
$Loc \in \mathbb{P}\ \mathbb{Z}$

A11++

theorem disabled rule dlLocsOfApplSubgoal

$$\begin{array}{l} \forall f : \langle\!|LOC : \mathbb{Z};\ SIZE : \mathbb{Z}|\!\rangle \nrightarrow \mathbb{P}\,\mathbb{Z}; \\ \quad x : \langle\!|LOC : \mathbb{Z};\ SIZE : \mathbb{Z}|\!\rangle;\ y : \mathbb{P}\,\mathbb{Z} \mid (x, y) \in f \bullet \\ \qquad (f(x) = y) \Leftrightarrow true \end{array}$$

theorem disabled rule dlLocsApplSubgoal

$$\begin{array}{l} \forall f : \mathbb{P}\,(\langle\!|LOC : \mathbb{Z};\ SIZE : \mathbb{Z}|\!\rangle) \nrightarrow \mathbb{P}\,\mathbb{Z}; \\ \quad x : \mathbb{P}\,(\langle\!|LOC : \mathbb{Z};\ SIZE : \mathbb{Z}|\!\rangle);\ y : \mathbb{P}\,\mathbb{Z} \mid (x, y) \in f \bullet \\ \qquad (f(x) = y) \Leftrightarrow true \end{array}$$

A11+

theorem disabled rule lLocsOfProp
$\forall p : Piece \bullet locs\_of\ (p) = p.LOC \mathrel{..} p.LOC + p.SIZE - 1$

theorem frule fFree1MaxElemType
$f \in Free1 \Rightarrow f \in \mathbb{P}\,(\langle\!|LOC : \mathbb{Z};\ SIZE : \mathbb{Z}|\!\rangle)$

A3

theorem grule gFree1Type
$Free1 \in \mathbb{P}\,(\mathbb{P}\,(\langle\!|LOC : \mathbb{Z};\ SIZE : \mathbb{Z}|\!\rangle))$

A3

theorem rule lLocsResMaxType
$\forall f : Free1 \bullet locs\,(f) \in \mathbb{P}\ \mathbb{Z}$

A11+A1

theorem rule lLocsOfResMaxType
$\forall p : Piece \bullet locs\_of\,(p) \in \mathbb{P}\,\mathbb{Z}$

A11+

theorem disabled rule dlFree1BigcupSubsumes
$\forall f : Free1 \bullet \bigcup\{p : f \bullet (locs\_of\ p)\} =$
$\bigcup\{q : Piece \mid q \in f \bullet (locs\_of\ q)\}$

theorem disabled rule lLocsProp
$\forall f : Free1 \bullet locs\,(f) = \bigcup\{p : f \bullet locs\_of\,(p)\}$

A6

theorem disabled rule dlInFree1
$f \in Free1 \Leftrightarrow (f \in \mathbb{P}\ Piece \land f \in \mathbb{F}\ Piece \land \mathbf{inv} - \mathbf{Free}\,(f))$

A7: add one for invFree as well

theorem disabled rule dlInInvFree1
$\mathbf{inv} - \mathbf{Free}\,(fr) \Leftrightarrow (fr \in \mathbb{P}\ Piece \land \forall p1, p2 : fr \bullet$
$p1 = p2 \lor$
$(locs\_of\,(p1) \cap locs\_of\,(p2) = \varnothing \land$
$p1.LOC + p1.SIZE \neq p2.LOC))$

A5

theorem rule lFree1ReductionType
$\forall f : Free1;\ p : Piece \bullet f \setminus \{p\} \in Free1$

```
A5
VO of lemma: not matcthing
begin{theorem}{lLocsOfWithinLocsVO}
\forall f: Free1 @ \forall p: Piece | p \in f @ \locsOf~(p) \in \power~(locs~(f))
end{theorem}
```

theorem rule lLocsOfWithinLocs
$\forall f : Free1 \bullet \forall p : f \bullet locs\_of\,(p) \in \mathbb{P}\,(locs\,(f))$

——— A5

theorem rule lLocsDistUnitDiff
$\forall f : Free1 \bullet \forall p : f \bullet locs\,(f \setminus \{p\}) = locs\,(f) \setminus locs\_of\,(p)$

```
This is the general lemma, but it relies on another lemma about $locs$ of unit sets.
begin{theorem}{rule lLocsDistDiff}
\forall f, g: Free1 @ locs~(f \setminus g) = locs~(f) \setminus locs~(g)
end{theorem}

begin{theorem}{rule lLocsUnit}
\forall p: Piece @ locs~(\{p\}) = \locsOf~(p)
end{theorem}
---------

A6: there is something wrong/not effective with the lemma below
begin[disabled]{theorem}{rule dlInFree1Elem}
\forall fr: Free1 @ p \in fr \iff (p \in Piece \land \\
\forall p1: fr @ \\
    \t1 p = p1 \lor \\
    \t1 \locsOf~(p) \cap \locsOf~(p1) = \emptyset \lor \\
    \t1 p.LOC + p.SIZE \neq p1.LOC )
end{theorem}
A9: try it again, slightly different
begin[disabled]{theorem}{rule dlInFree1Elem}
\forall fr: Free1 @ p \in fr \iff (p \in Piece \land \\
\forall q: fr @ \\
    \t1 p = q \lor \\
    \t1 \locsOf~(p) \cap \locsOf~(q) = \emptyset \lor \\
    \t1 p.LOC + p.SIZE \neq q.LOC )
end{theorem}

A10: useful for extracting the type for q, but it's not quite right shape; added explicitly the type for q
begin[disabled]{theorem}{rule dlInFree1Elem}
\forall fr: Free1 @ p \in fr \iff (p \in Piece \land \\
\forall q: Piece | q \in fr @ \\
    \t1 p = q \lor \\
    \t1 \locsOf~(p) \cap \locsOf~(q) = \emptyset \lor \\
    \t1 p.LOC + p.SIZE \neq q.LOC )
end{theorem}
```

A11

theorem disabled rule dNEW1AuxLemma1

$$\forall fr : Free1;\ p : Piece;\ size : \mathbb{Z} \mid p \in fr \land p.SIZE \geq size \geq 0 \bullet$$
$$\theta(Piece[LOC := p.LOC, SIZE := size]) \in fr$$

6 VCs

PieceFSBSig
Piece
Piece

$Heap1FSBSig$

$$\begin{array}{|l}
\quad Heap1 \\
\hline
Heap1
\end{array}$$

$NEW1FSBSig$

$$\begin{array}{|l}
\quad Heap1 \\
req? : \mathbb{N} \\
\hline
\exists\, p : Piece \bullet p \in free1 \land p.SIZE \geq req?
\end{array}$$

$DISPOSE1FSBSig$

$$\begin{array}{|l}
\quad Heap1 \\
ret? : Piece \\
\hline
locs_of\ (ret?) \cap locs\ (free1) = \varnothing
\end{array}$$

theorem Piece_ vc_ fsb_ state
$\exists\, PieceFSBSig \mid true \bullet true$

theorem Heap1_ vc_ fsb_ state
$\exists\, Heap1FSBSig \mid true \bullet true$

theorem NEW1_ vc_ fsb_ pre
$\forall\, NEW1FSBSig \mid true \bullet \mathbf{pre}\ NEW1$

theorem DISPOSE1_ vc_ fsb_ pre
$\forall\, DISPOSE1FSBSig \mid true \bullet \mathbf{pre}\ DISPOSE1$

7 Proofs

7.1 Domain checks

proof[*lLocsOfIsTotal*]
■

proof[$invFree\$domainCheck$]
  prove by reduce;
  ∎

```
A1: added lemma: lLocsOfIsTotal (useful in various domain checks below)
prove by reduce;
apply inDom;
prove;
invoke \locsOf;
prove;
apply inPower;
instantiate e == p1;
instantiate e == p2;
with enabled (Piece\$member) prove by reduce;
A0: added lemma about gLocsOfPFunType
apply pfunAppliesTo;
with enabled (\locsOf) prove by reduce;
added lemma: gLocsOfPFunType
```

proof[$locs\$domainCheck$]
  prove by reduce;
  ∎

proof[$NEW1\$domainCheck$]
  prove by reduce;
  ∎

proof[$DISPOSE1\$domainCheck$]
  prove by reduce;
  ∎

proof[$Piece\_vc\_fsb\_state$]
  instantiate $LOC == 0, SIZE == 0$;
  prove by reduce;
  ∎

proof[$Heap1\_vc\_fsb\_state$]
  instantiate $free1 == \varnothing$;
  with enabled ($Free1$, **inv − Free** _) *prove by reduce*;
  ∎

proof[$NEW1\_\ vc\_\ fsb\_\ pre$]

$\quad$ *prove by reduce*;

$\quad$ *instantiate* $free1' == free1$

$\quad$ $\setminus \{ \theta\ Piece[LOC := p.LOC,\ SIZE := req?] \}$,

$\quad$ $res! == \theta\ Piece[LOC := p.LOC,\ SIZE := req?]$;

$\qquad$ *with enabled* ($Piece\$member$) *prove by reduce*;

$\quad$ *apply* $lLocsOfWithinLocs$;

$\quad$ *apply* $lLocsDistUnitDiff$;

$\quad$ *prove by*

$\quad$ *rewrite*;

split $\theta\ (Piece\ [SIZE := req?]) \in free1$;

rewrite;

rearrange;

use $dNEW1AuxLemma1[fr := free1,\ p := p,\ size := req?]$;

prove by reduce;

■

proof[$DISPOSE1\_\ vc\_\ fsb\_\ pre$]

prove by reduce;

■

proof[$gFree1Type$]

with enabled ($Free1$, $\mathbf{inv - Free}\ \_$) *prove by reduce*;

■

proof[$gLocsPFunType$]

invoke locs;

prove by reduce;

■

proof[$lLocsResMaxType$]

■

proof[*lLocsOfWithinLocs*]
apply lLocsProp;
rewrite;
apply inPowerBigcup ;
prove;
instantiate $p\_\_0 \;==\; p$;
prove;
■

```
A2: added extra frule for fFree1ElemType (as Piece and not max)
apply lLocsProp;
rewrite;
apply inPowerBigcup ;
prove;
instantiate p\_\_0 == p;
prove;
apply inPower;
prove;
apply lLocsOfResMaxType;
rewrite;
use dlInFree1ElemType[fr := f, p := p\_\_0];
prove;

A1: start from locs -> works!;;; added lemma : lLocsOfResMaxType
apply lLocsProp;
rewrite;
apply inPowerBigcup ;
prove;
instantiate p\_\_0 == p;
prove;
apply inPower;
prove;
apply lLocsOfResMaxType;
A0: start from locsOf -> nope...
apply lLocsOfProp;
prove;
use dlInFree1ElemType[fr := f, p := p];
with enabled (Piece\$member) prove;
apply inPower;
prove;
apply lLocsProp;
rewrite;
apply inBigcup;
rewrite;
cases;
instantiate B == \locsOf~p;
prove;
apply inPower;
prove;
apply inPowerBigcup;
prove;
```

proof[$dlInFree1ElemType$]
  prove;
  ■

proof[$fFree1ElemType$]
  *invoke Free*1;
  prove;
  ■

proof[$gLocType$]
  invoke Loc;
  prove;
  ■

proof[$dlLocsOfApplSubgoal$]
  $use\ pairInFunction[(\lblot\ LOC : \mathbb{Z};\ \ SIZE : \mathbb{Z} \rblot),\ \mathbb{P}\ \mathbb{Z}]$;
  prove by rewrite;
  ■

proof[$lLocsOfProp$]
  apply dlLocsOfApplSubgoal;
  rewrite;
  invoke locs_of;
  apply Piece$*member*;
  prove by reduce;
  ■

proof[$dlLocsApplSubgoal$]
  $use\ pairInFunction[\mathbb{P}\,(\lblot\ LOC : \mathbb{Z};\ \ SIZE : \mathbb{Z} \rblot),\ \mathbb{P}\ \mathbb{Z}]$;
  prove by rewrite;
  ■

proof[$fFree1MaxElemType$]
  *invoke Free*1;
  prove;
  ■

proof[*dlFree1BigcupSubsumes*]

```
split f = {};
cases;
prove;
 apply extensionality;
 prove;
 apply inBigcup;
 prove;
next;
apply extensionality ;
 prove;
 apply inBigcup;
 prove;
 split ¬ {p__1 : f • locs_of p__1 } ∈ ℙ (ℙ ℤ);
 cases;
 apply inPower;
 instantiate e == locs_of x;
 prove;
next;
cases;
split ¬ x__0 ∈ ℤ ;
cases;
prove;
next;
prove;
instantiate B__0 == B;
prove;
instantiate p__0 == p;
rewrite;
next;
split ¬ y__0 ∈ ℤ ;
cases;
prove;
next;
prove;
instantiate B == B__0;
prove;
instantiate p__0 == p;
rewrite;
next;
```

■

proof[*lLocsProp*]
apply dlLocsApplSubgoal;
rewrite;
invoke locs;
prove;
*apply dlFree*1*BigcupSubsumes*;
prove;
∎

```
A1: adding fFree1ElemType, fFree1MaxElemType made all the different!
apply dlLocsApplSubgoal;
rewrite;
apply bigcupInPower;
rewrite;
apply inPower to predicate \{p: f @ \locsOf p \} \in \power (\power \num);
prove;
apply lLocsOfResMaxType;
rewrite;
use dlInFree1ElemType[fr := f, p := p];
prove;
invoke locs;
prove;
apply dlFree1BigcupSubsumes;
prove;
A0?
apply dlLocsApplSubgoal;
rewrite;
apply bigcupInPower;
rewrite;
apply inPower to predicate \{p: f @ \locsOf p \} \in \power (\power \num);
prove;
apply lLocsOfResMaxType;
rewrite;
use dlInFree1ElemType[fr := f, p := p];
prove;
invoke locs;
%ahhh/... that's annoying! -- added general lemma about big union
prove by reduce;
apply extensionality;
prove;
apply inBigcup;
prove;
\end{zproof}
% added: fFree1MaxElemtType
```

proof[*dlInInvFree*1]
invoke ($\mathbf{inv} - \mathbf{Free}_{\_}$);
prove;
∎

proof[*dlInFree*1]
*invoke Free*1;
prove;
■

proof[*lFree*1*ReductionType*]
*apply dlInFree*1;
prove;
*apply dlInInvFree*1;
prove;
instantiate $p1\_\_0 == p1$, $p2\_\_0 == p2$;
prove;
■

proof[$lLocsDistUnitDiff$]

$apply\ lLocsProp$;
$prove$;
$apply\ extensionality$;
$prove$;
$apply\ inBigcup$;
$prove$;
$cases$;
$split\ \neg\ x\ \in\ \mathbb{Z}$;
$cases$;
$prove$;
$next$;
$rewrite$;
$instantiate\ B\_\_0\ ==\ B$;
$prove$;
$instantiate\ p\_\_1\ ==\ p\_\_0$;
$prove$;
$apply\ dlInFree1$;
$apply\ dlInInvFree1$;
$instantiate\ p1\ ==\ p,\ p2 ==\ p\_\_0$;
$prove$;
$apply\ extensionality\ to\ predicate\ \ locs\_of\ \ p\ \cap\ locs\_of\ \ p\_\_0\ =\ \{\}$;
$prove$;
$instantiate\ x\_\_0\ ==\ x$;
$prove$;
$next$;
$split\ \neg\ y\ \in\ \mathbb{Z}$;
$cases$;
$prove$;
$next$;
$rewrite$;
$instantiate\ B\ ==\ B\ \_\_0$;
$prove$;
$instantiate\ p\_\_1\ ==\ p\_\_0$;
$prove$;
$next$;
■

proof[$dNEW1AuxLemma1$]
apply $dlInFree1$;
apply $dlInInvFree1$;
prove;
instantiate $p1 == p, p2 == \theta\ Piece[LOC := p.LOC, SIZE := size]$;
with enabled $(Piece\$member)$ *prove*;
apply extensionality to predicate $locs\_of\ (\theta\ Piece) \cap locs\_of\ (\theta\ (Piece\ [SIZE := size])) = \{\}$;
prove;
split $SIZE = size$;
cases;
prove;
next;
rewrite;
apply $lLocsOfProp$;
reduce;
instantiate $x == LOC$;
prove;
split $size = 0$;
rewrite;

■

```
AO: hum... extensionality latter wasn't orking
    split p.SIZE = size;
    cases;
     with enabled (Piece\$member) prove;
    next;
apply dlInFree1;
apply dlInInvFree1;
prove;
instantiate p1 == p, p2 == \theta Piece[LOC := p.LOC, SIZE := size];
with enabled (Piece\$member) prove;
apply lLocsOfProp to expression \locsOf (\theta Piece);
apply lLocsOfProp to expression \locsOf (\theta (Piece [SIZE := size]));
reduce;
apply extensionality to predicate (LOC \upto (\negate 1 + (size + LOC))) \cap (LOC \upto (\negate 1 + (LOC +
prenex;
rewrite;
split \theta (Piece [SIZE := size]) \in fr;

simplify;
with normalization rewrite;
```