A proof of Doug Lea's memory manager

John Wickerson

November 30, 2011

Chapter 1

Glossary of macros, typedefs and minor routines

MALLOC_ALIGNMENT	$= 8$
MAX_SIZE_T	$= FFFF\ FFFF_h$
SIZE_T_SIZE	$= 4$
SIZE_T_BITSIZE	$= 32$
SIZE_T_ZERO	$= 0$
SIZE_T_ONE	$= 1$
SIZE_T_TWO	$= 2$
SIZE_T_FOUR	$= 4$
TWO_SIZE_T_SIZES	$= 8$
FOUR_SIZE_T_SIZES	$= 16$
SIX_SIZE_T_SIZES	$= 24$
HALF_MAX_SIZE_T	$= 7FFF\ FFFF_h$
CHUNK_ALIGN_MASK	$= 111_b$
mchunk	$=$ struct malloc_chunk
mchunkptr	$=$ mchunk*
sbinptr	$=$ mchunk*
bindex_t	$=$ unsigned int
binmap_t	$=$ unsigned int
flag_t	$=$ unsigned int
MCHUNK_SIZE	$= 16$
CHUNK_OVERHEAD	$= 4$
MIN_CHUNK_SIZE	$= 16$
chunk2mem(p)	$= \mathtt{p} + 8$
mem2chunk(mem)	$= \mathtt{mem} - 8$
MAX_REQUEST	$= 2^{32} - 2^6$
MIN_REQUEST	$= 11$
pad_request(req)	$= \lceil \mathtt{req} + 4 \rceil_8$
request2size(req)	$= \max\{16, \lceil \mathtt{req} + 4 \rceil_8\}$
PINUSE_BIT	$= 1_b$
CINUSE_BIT	$= 10_b$
FLAG4_BIT	$= 100_b$
INUSE_BITS	$= 11_b$
FLAG_BITS	$= 111_b$
cinuse(p)	$= [\mathtt{p}_{[1]}] == 1$
pinuse(p)	$= [\mathtt{p}_{[0]}] == 1$
is_inuse(p)	$=$ is_mmapped(p) $\vee$ cinuse(p)
is_mmapped(p)	$= [\mathtt{p}_{[1,0]}] == 00$
chunksize(p)	$= [(\mathtt{p} + 1)_{[31..3]}000]$
$\{\mathtt{p}_{[0]} \mapsto \_\}$ clear_pinuse(p)	$\{\mathtt{p}_{[0]} \mapsto 0\}$
chunk_plus_offset(p,s)	$= p + s$
chunk_minus_offset(p,s)	$= p - s$
next_chunk(p)	$= next(\mathtt{p})$
prev_chunk(p)	$= prev(\mathtt{p})$
next_pinuse(p)	$= flags(next(\mathtt{p})) = \_\blacktriangle$
get_foot(p,s)	$= prev\_foot(\mathtt{p} + \mathtt{s})$
$\{prev\_foot(\mathtt{p} + \mathtt{s}) = \_\}$ set_foot(p,s)	$\{prev\_foot(\mathtt{p} + \mathtt{s}) = \mathtt{s}\}$
$\{size(\mathtt{p}) = \_ \wedge flags(\mathtt{p}) = \_\_ \wedge prev\_foot(\mathtt{p} + \mathtt{s}) = \_\}$ set_size_and_pinuse_of_free_chunk(p,s)	$\{size(\mathtt{p}) = \mathtt{s} \wedge flags(\mathtt{p}) = \triangledown\blacktriangle \wedge prev\_foot(next(\mathtt{p})) = \mathtt{s}\}$
$\{size(\mathtt{p}) = \_ \wedge flags(\mathtt{p}) = \_\_ \wedge prev\_foot(\mathtt{p} + \mathtt{s}) = \_ \wedge flags(\mathtt{p} + \mathtt{s}) = \_\_\}$ set_free_with_pinuse(p,s,n)	$\{size(\mathtt{p}) = \mathtt{s} \wedge flags(\mathtt{p}) = \triangledown\blacktriangle \wedge prev\_foot(next(\mathtt{p})) = \mathtt{s} \wedge flags(next(\mathtt{p})) = \_\triangle\}$
tchunk	$=$ malloc_tree_chunk
tchunkptr	$=$ tchunk*
tbinptr	$=$ tchunk*
leftmost_child(t)	$= \begin{cases} child_0(*\mathtt{t}) & \text{if } child_0(*\mathtt{t}) \neq 0 \\ child_1(*\mathtt{t}) & \text{otherwise} \end{cases}$
NSMALLBINS	$= 32$
NTREEBINS	$= 32$
SMALLBIN_SHIFT	$= 3$
SMALLBIN_WIDTH	$= 8$
TREEBIN_SHIFT	$= 8$
MIN_LARGE_SIZE	$= 256$
MAX_SMALL_SIZE	$= 255$
MAX_SMALL_REQUEST	$= 244$
mstate	$=$ struct malloc_state
mparams	$=$ struct malloc_params
is_small(s)	$= \mathtt{s} < 256$
small_index(s)	$= \lfloor \mathtt{s}/8 \rfloor$
small_index2size(i)	$= 8 \times \mathtt{i}$
MIN_SMALL_INDEX	$= 2$
$\{\mathtt{smallbins}[2\mathtt{i}+2] \mapsto C_1 * \mathtt{smallbins}[2\mathtt{i}+3] \mapsto C_2\}$ x := smallbin_at(M,i)	$\{\mathtt{x.fd} \mapsto C_1 * \mathtt{x.bk} \mapsto C_2\}$
treebin_at(M,i)	$=$ treebins[i]
$\{\mathtt{I} = \_\}$ compute_tree_index(S,I)	$\left\{\mathtt{I} = \begin{cases} 0 & \text{if } \mathtt{S} < 256 \\ 31 & \text{if } \mathtt{S} > 2^{24} \\ 2(\log_2 \llbracket \mathtt{S} \rrbracket - 8) & \text{if } 0 \leq \{\!\{\mathtt{S}\}\!\} < \frac{1}{2}\llbracket \mathtt{S} \rrbracket \\ 2(\log_2 \llbracket \mathtt{S} \rrbracket - 8) + 1 & \text{if } \frac{1}{2}\llbracket \mathtt{S} \rrbracket \leq \{\!\{\mathtt{S}\}\!\} < \llbracket \mathtt{S} \rrbracket \end{cases}\right\}$
bin_for_tree_index(i)	$= \begin{cases} 31 & \text{if } \mathtt{i} = 31 \\ \lfloor \mathtt{i}/2 \rfloor + 6 & \text{otherwise} \end{cases}$
leftshift_for_tree_index(i)	$= \begin{cases} 0 & \text{if } \mathtt{i} = 31 \\ 25 - \lfloor \mathtt{i}/2 \rfloor & \text{otherwise} \end{cases}$
minsize_for_tree_index(i)	$= \begin{cases} 2 \ll (\lfloor \mathtt{i}/2 \rfloor + 7) & \text{if i even} \\ 3 \ll (\lfloor \mathtt{i}/2 \rfloor + 7) & \text{if i odd} \end{cases}$
idx2bit(i)	$= 1 \ll \mathtt{i}$
$\{\mathtt{smallmap}[\mathtt{i}] = \_\}$ mark_smallmap(M,i)	$\{\mathtt{smallmap}[\mathtt{i}] = 1\}$
$\{\mathtt{smallmap}[\mathtt{i}] = \_\}$ clear_smallmap(M,i)	$\{\mathtt{smallmap}[\mathtt{i}] = 0\}$
smallmap_is_marked(M,i)	$=$ smallmap[i] $= 1$
$\{\mathtt{treemap}[\mathtt{i}] = \_\}$ mark_treemap(M,i)	$\{\mathtt{treemap}[\mathtt{i}] = 1\}$
$\{\mathtt{treemap}[\mathtt{i}] = \_\}$ clear_treemap(M,i)	$\{\mathtt{treemap}[\mathtt{i}] = 0\}$
treemap_is_marked(M,i)	$=$ treemap[i] $= 1$
least_bit(x)	$= \begin{cases} \mathbf{0}\,\overset{i}{1}\,\mathbf{0} & \text{if } \mathtt{x}_i = 1 \wedge \forall j < i.\, \mathtt{x}_j = 0\} \\ \mathbf{0} & \text{if x=0} \end{cases}$
left_bits(x)	$= \begin{cases} \mathbf{1}\,\overset{i}{0}\,\mathbf{0} & \text{if } \mathtt{x}_i = 1 \wedge \forall j < i.\, \mathtt{x}_j = 0\} \\ \mathbf{0} & \text{if } \mathtt{x} = 0 \end{cases}$
same_or_left_bits(x)	$= \begin{cases} \mathbf{1}\,\overset{i}{1}\,\mathbf{0} & \text{if } \mathtt{x}_i = 1 \wedge \forall j < i.\, \mathtt{x}_j = 0\} \\ \mathbf{0} & \text{if } \mathtt{x} = 0 \end{cases}$
$\{\mathtt{I} = \_\}$ compute_bit2idx(X,I)	$\{\mathtt{X} \neq 0 \Rightarrow \mathtt{I} = \log_2 \mathtt{X}\}$
$\{p\}$ mark_inuse_foot(M,p,s)	$\{p\}$
$\{size(\mathtt{p}) = \_ \wedge flags(\mathtt{p}) = \_P \wedge flags(\mathtt{p} + \mathtt{s}) = C\_\}$ set_inuse(M,p,s)	$\{size(\mathtt{p}) = \mathtt{s} \wedge flags(\mathtt{p}) = \blacktriangledown P \wedge flags(next(\mathtt{p})) = C\blacktriangle\}$
$\{size(\mathtt{p}) = \_ \wedge flags(\mathtt{p}) = \_\_ \wedge flags(\mathtt{p} + \mathtt{s}) = C\_\}$ set_inuse_and_pinuse(M,p,s)	$\{size(\mathtt{p}) = \mathtt{s} \wedge flags(\mathtt{p}) = \blacktriangledown\blacktriangle \wedge flags(next(\mathtt{p})) = C\blacktriangle\}$
$\{size(\mathtt{p}) = \_ \wedge flags(\mathtt{p}) = \_\_\}$ set_inuse_and_pinuse_of_inuse_chunk(M,p,s)	$\{size(\mathtt{p}) = \mathtt{s} \wedge flags(\mathtt{p}) = \blacktriangledown\blacktriangle\}$

Chapter 2

State

Shorthand:

$$
\begin{array}{rcl}
|i| & \stackrel{\text{def}}{=} & \{8i\} \\
\|i\| & \stackrel{\text{def}}{=} & \texttt{compute\_tree\_index}^{-1}(i) \\
\mathsf{w} & \stackrel{\text{def}}{=} & 4 \\
x \uplus y & \stackrel{\text{def}}{=} & \begin{cases} x \cup y & \text{if } x \cap y = \{\} \\ \text{undefined} & \text{otherwise} \end{cases} \\
x \uplus\!\!\!- \, y & \stackrel{\text{def}}{=} & \begin{cases} x - y & \text{if } y \subseteq x \\ \text{undefined} & \text{otherwise} \end{cases} \\
a \,{}'_{r}\, b & \stackrel{\text{def}}{=} & \begin{cases} a/b & \text{if } b \text{ divides } a \\ \text{undefined} & \text{otherwise} \end{cases}
\end{array}
$$

Predicates:

$$
\begin{array}{rcl}
x \xmapsto{\mathsf{prevfoot}} s & \stackrel{\text{def}}{=} & x \mapsto s \\
x \xmapsto{\mathsf{size}} s & \stackrel{\text{def}}{=} & x + 1\mathsf{w} \mapsto_{[31..3]} s \,{}'_{r}\, 8 \\
x \xmapsto{\mathsf{pinuse}} b & \stackrel{\text{def}}{=} & x + 1\mathsf{w} \mapsto_{[0]} b \\
x \xmapsto{\mathsf{cinuse}} b & \stackrel{\text{def}}{=} & x + 1\mathsf{w} \mapsto_{[1]} b \\
x \xmapsto{\mathsf{fd}} y & \stackrel{\text{def}}{=} & x + 2\mathsf{w} \mapsto y \\
x \xmapsto{\mathsf{bk}} y & \stackrel{\text{def}}{=} & x + 3\mathsf{w} \mapsto y \\
\mathit{ublock}(x, y, B) & \stackrel{\text{def}}{=} & \mathsf{let}\ s = y - x\ \mathsf{in}\ \exists n.\, B = \{x + 2\mathsf{w} \mapsto_{\mathsf{u}} n\mathsf{w}\}\ *\ (n+1)\mathsf{w} = s \\
& & *\ \tfrac{1}{2}(x \xmapsto{\mathsf{size}} s)\ *\ y \xmapsto{\mathsf{pinuse}} 0\ *\ x \xmapsto{\mathsf{cinuse}} 0 \\
& & *\ s \geq 4\mathsf{w}\ *\ \mathop{\ast}_{i=4}^{s \,{}'_{r}\, \mathsf{w}}. x + i\mathsf{w} \mapsto \_\ *\ y \xmapsto{\mathsf{prevfoot}} s \\
\mathit{ablock}(x, y, B) & \stackrel{\text{def}}{=} & \mathsf{let}\ s = y - x\ \mathsf{in}\ \exists n.\, B = \{x + 2\mathsf{w} \mapsto_{\mathsf{a}} n\mathsf{w}\}\ *\ (n+1)\mathsf{w} \leq s \\
& & *\ x \xmapsto{\mathsf{size}} s\ *\ y \xmapsto{\mathsf{pinuse}} 1\ *\ x \xmapsto{\mathsf{cinuse}} 1 \\
& & *\ s \geq 4\mathsf{w}\ *\ \mathop{\ast}_{i=n+2}^{s \,{}'_{r}\, \mathsf{w}+1}. x + i\mathsf{w} \mapsto \_ \\
\mathit{block} & \stackrel{\text{def}}{=} & \mathit{ublock} \vee \mathit{ablock} \\
\mathit{bin}(S, x, U) & \stackrel{\text{def}}{=} & (U = \{\}\ *\ x \xmapsto{\mathsf{fd}} \_\ *\ x \xmapsto{\mathsf{bk}} \_) \\
& & \vee\, (\exists y.\, x \xmapsto{\mathsf{fd}} y\ *\ y \xmapsto{\mathsf{bk}} x\ *\ (\mathit{bnode}\ S)^*(y, x, U)) \\
\mathit{bnode}\ S\ (x, y, U) & \stackrel{\text{def}}{=} & \exists s.\, x \xmapsto{\mathsf{fd}} y\ *\ y \xmapsto{\mathsf{bk}} x\ *\ U = \{x + 2\mathsf{w} \mapsto s - 1\mathsf{w}\}\ *\ \tfrac{1}{2}(x \xmapsto{\mathsf{size}} s)\ *\ s \in S \\
\mathit{sorted}(L, \sqsubseteq) & \stackrel{\text{def}}{=} & \forall i, j.\, i \leq j \Rightarrow L(i) \sqsubseteq L(j) \\
\mathit{coalesced}(B) & \stackrel{\text{def}}{=} & \exists L.\, \operatorname{ran} L = B\ *\ \mathit{sorted}(L, \leq_1)\ *\ \nexists i.\, (L(i))_3 = (L(i+1))_3 = \mathsf{u} \\
\mathit{arena}(B, t) & \stackrel{\text{def}}{=} & \mathit{coalesced}(B \uplus t)\ *\ \texttt{start} \xmapsto{\mathsf{pinuse}} 1\ *\ \texttt{start} \xmapsto{\mathsf{prevfoot}} \_ \\
& & *\ \mathit{block}^*(\texttt{start}, \texttt{top}, B) \\
\mathit{smallbin}_i(U) & \stackrel{\text{def}}{=} & i \in [0\,..\,32)\ *\ \mathit{bin}(|i|, \texttt{smallbin} + 2i\mathsf{w}, U)\ *\ \texttt{smallmap}_{[i]} = (U \neq \{\}) \\
\mathit{treebin}_i(U) & \stackrel{\text{def}}{=} & i \in [0\,..\,32)\ *\ \mathit{bin}(\|i\|, \texttt{treebins} + i\mathsf{w}, U)\ *\ \texttt{treemap}_{[i]} = (U \neq \{\}) \\
\mathit{victim}(v) & \stackrel{\text{def}}{=} & (\texttt{dvsize} = 0\ *\ v = \{\}) \vee \\
& & (\texttt{dvsize} \,{}'_{r}\, 8 \in [2\,..\,32)\ *\ v = \{\texttt{dv} + 2\mathsf{w} \mapsto_{\mathsf{u}} \texttt{dvsize} - 1\mathsf{w}\} \\
& & *\ \texttt{dv} \xmapsto{\mathsf{fd}} \_\ *\ \texttt{dv} \xmapsto{\mathsf{bk}} \_\ *\ \tfrac{1}{2}(\texttt{dv} \xmapsto{\mathsf{size}} \texttt{dvsize})) \\
\mathit{topchunk}(t) & \stackrel{\text{def}}{=} & t = \{\texttt{top} + 2\mathsf{w} \mapsto_{\mathsf{u}} \texttt{topsize} - 1\mathsf{w}\}\ *\ \texttt{top} \xmapsto{\mathsf{cinuse}} 0\ *\ \texttt{top} \xmapsto{\mathsf{size}} \texttt{topsize} \\
& & *\ \texttt{topsize} \,{}'_{r}\, 8 \in [0\,..\,2^{29} - 8]\ *\ \mathop{\ast}_{i=2}^{\texttt{topsize}/\mathsf{w}}. \texttt{top} + i\mathsf{w} \mapsto \_ \\
\mathit{state}(A) & \stackrel{\text{def}}{=} & \exists \{U_i \mid i \in [0\,..\,64)\}, v, t.\, \mathit{arena}(A_{\mathsf{a}} \uplus (\biguplus_{i=0}^{64}. U_i)_{\mathsf{u}} \uplus v, t) \\
& & *\ \mathop{\ast}_{i=0}^{32}. \mathit{smallbin}_i(U_i)\ *\ \mathop{\ast}_{i=0}^{32}. \mathit{treebin}_i(U_{i+32})\ *\ \mathit{victim}(v)\ *\ \mathit{topchunk}(t)
\end{array}
$$

Lemma 1. *The assertion*

$$\mathit{block}(x, y, B_1)\ *\ \mathit{ablock}(y, z, B_2)\ *\ \mathit{coalesced}(B_1 \uplus B_2 \uplus B_3)$$

implies

$$\mathit{ublock}(x, y, B_1)\ *\ \mathit{ablock}(y, z, B_2)\ *\ \mathit{coalesced}(B_1 \uplus B_2 \uplus B_3).$$

Chapter 3

Auxiliary operations

3.1 `set_inuse_and_pinuse`

TODO: We need `s` to be a multiple of 8 (or else return $\lfloor \mathtt{s} \rfloor_8$).

Specification:

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} \_ \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{pinuse}}{\longmapsto} \_ \right\}$$

```
set_inuse_and_pinuse(M,p,s)
```

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \mathtt{s} \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} 1 \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} 1 \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{pinuse}}{\longmapsto} 1 \right\}$$

Verification:

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} \_ \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{pinuse}}{\longmapsto} \_ \right\}$$

```
p->head = (s|PINUSE_BIT|CINUSE_BIT);
```

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \mathtt{s} \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} 1 \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} 1 \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{pinuse}}{\longmapsto} \_ \right\}$$

```
((mchunkptr)(((char*)p) + s))->head |= PINUSE_BIT;
```

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \mathtt{s} \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} 1 \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} 1 \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{pinuse}}{\longmapsto} 1 \right\}$$

3.2 `set_size_and_pinuse_of_free_chunk`

TODO: We need `s` to be a multiple of 8.

Specification:

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} \_ \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{prevfoot}}{\longmapsto} \_ \right\}$$

```
set_size_and_pinuse_of_free_chunk(p,s)
```

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \mathtt{s} \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} 1 \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} 0 \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{prevfoot}}{\longmapsto} \mathtt{s} \right\}$$

Verification:

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} \_ \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{prevfoot}}{\longmapsto} \_ \right\}$$

```
p->head = (s|PINUSE_BIT);
```

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \mathtt{s} \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} 1 \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} 0 \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{prevfoot}}{\longmapsto} \_ \right\}$$

```
set_foot(p,s);
```

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \mathtt{s} \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} 1 \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} 0 \ast \mathtt{p}+\mathtt{s} \overset{\mathsf{prevfoot}}{\longmapsto} \mathtt{s} \right\}$$

3.3 `set_size_and_pinuse_of_inuse_chunk`

TODO: We need `s` to be a multiple of 8 (or else return $\lfloor \mathtt{s} \rfloor_8$).

Specification:

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} \_ \right\}$$

```
set_size_and_pinuse_of_inuse_chunk(M,p,s)
```

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \mathtt{s} \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} 1 \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} 1 \right\}$$

Verification:

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} \_ \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} \_ \right\}$$

```
p->head = (s|PINUSE_BIT|CINUSE_BIT);
```

$$\left\{ \mathtt{p} \overset{\mathsf{size}}{\longmapsto} \mathtt{s} \ast \mathtt{p} \overset{\mathsf{pinuse}}{\longmapsto} 1 \ast \mathtt{p} \overset{\mathsf{cinuse}}{\longmapsto} 1 \right\}$$

3.4 `insert_small_chunk`

Specification:

$$\left\{ \tfrac{1}{2}(\mathtt{P} \overset{\mathsf{size}}{\longmapsto} \mathtt{S}) \ast \mathtt{P} \overset{\mathsf{fd}}{\longmapsto} \_ \ast \mathtt{P} \overset{\mathsf{bk}}{\longmapsto} \_ \ast \mathit{smallbin}_{\lfloor \mathtt{S}/8 \rfloor}(U) \right\}$$

```
#define insert_small_chunk(M, P, S) {\
  bindex_t I = small_index(S);\
  mchunkptr B = smallbin_at(M, I);\
  mchunkptr F = B;\
  assert(S >= MIN_CHUNK_SIZE);\
  if (!smallmap_is_marked(M, I))\
    mark_smallmap(M, I);\
  else if (RTCHECK(ok_address(M, B->fd)))\
    F = B->fd;\
  else {\
    CORRUPTION_ERROR_ACTION(M);\
  }\
  B->fd = P;\
  F->bk = P;\
  P->fd = F;\
  P->bk = B;\
}
```

$$\left\{ \mathit{smallbin}_{\lfloor \mathtt{S}/8 \rfloor}(U \uplus \{\mathtt{P}+2\mathtt{w} \mapsto \mathtt{S}-1\mathtt{w}\}) \right\}$$

3.5 unlink_small_chunk

Specification:

$\left\{ smallbin_{\lfloor \mathsf{S}/8 \rfloor}(U \uplus \{\mathsf{P} + 2\mathsf{w} \mapsto \mathsf{S} - 1\mathsf{w}\}) \right\}$

```
#define unlink_small_chunk(M, P, S) {\
  mchunkptr F = P->fd;\
  mchunkptr B = P->bk;\
  bindex_t I = small_index(S);\
  assert(P != B);\
  assert(P != F);\
  assert(chunksize(P) == small_index2size(I));\
  if (F == B)\
    clear_smallmap(M, I);\
  else if (RTCHECK((F == smallbin_at(M,I) || ok_address(M, F)) &&\
                   (B == smallbin_at(M,I) || ok_address(M, B)))) {\
    F->bk = B;\
    B->fd = F;\
  }\
  else {\
    CORRUPTION_ERROR_ACTION(M);\
  }\
}
```

$\left\{ \frac{1}{2}(\mathsf{P} \overset{\mathsf{size}}{\longmapsto} \mathsf{S}) \ * \ \mathsf{P} \overset{\mathsf{fd}}{\longmapsto} \_ \ * \ \mathsf{P} \overset{\mathsf{bk}}{\longmapsto} \_ \ * \ smallbin_{\lfloor \mathsf{S}/8 \rfloor}(U) \right\}$

3.6 unlink_first_small_chunk

Specification:

$$\left\{\begin{array}{l}\exists F.\,\mathtt{B} = \mathtt{smallbin} + 2\mathtt{Iw} \;*\; 0 \le \mathtt{I} < 32 \\ *\; \mathtt{B} \overset{\mathsf{fd}}{\longmapsto} \mathtt{P} \;*\; \mathtt{P} \overset{\mathsf{bk}}{\longmapsto} \mathtt{B} \;*\; \frac{1}{2}(\mathtt{P} \overset{\mathsf{size}}{\longmapsto} 8\mathtt{I}) \;*\; \mathtt{P} \overset{\mathsf{fd}}{\longmapsto} F \;*\; F \overset{\mathsf{bk}}{\longmapsto} \mathtt{P} \\ *\; (bnode\,|\mathtt{I}|)^*(F, \mathtt{B}, U) \;*\; \mathtt{smallmap}_{[\mathtt{I}]} = 1\end{array}\right\}$$

```
#define unlink_first_small_chunk(M, B, P, I) {\
  mchunkptr F = P->fd;\
  assert(P != B);\
  assert(P != F);\
  assert(chunksize(P) == small_index2size(I));\
  if (B == F)\
    clear_smallmap(M, I);\
  else if (RTCHECK(ok_address(M, F))) {\
    B->fd = F;\
    F->bk = B;\
  }\
  else {\
    CORRUPTION_ERROR_ACTION(M);\
  }\
}
```

$$\left\{\frac{1}{2}(\mathtt{P} \overset{\mathsf{size}}{\longmapsto} 8\mathtt{I}) \;*\; \mathtt{P} \overset{\mathsf{fd}}{\longmapsto} \_ \;*\; \mathtt{P} \overset{\mathsf{bk}}{\longmapsto} \_ \;*\; smallbin_{\mathtt{I}}(U)\right\}$$

3.7 replace_dv

Specification:

$$\left\{\begin{array}{l}\exists v, \{U_i \mid i \in [0,64)\}.\, \texttt{least\_addr} = 5 \;*\; \ast_{i=0}^{32}.\, treebin_i(U_{i+32}) \\ *\; arena(A_{\mathsf{a}} \uplus (\uplus_{i=0}^{64}.\, U_i)_{\mathsf{u}} \uplus v \uplus \{\mathsf{P} + 2\mathsf{w} \mapsto_{\mathsf{u}} \mathsf{S} - 1\mathsf{w}\}, t) \\ *\; \ast_{i=0}^{32}.\, smallbin_i(U_i) \;*\; victim(v) \;*\; \mathsf{P} \xmapsto{\mathsf{fd}} \_ \\ *\; \mathsf{P} \xmapsto{\mathsf{bk}} \_ \;*\; \frac{1}{2}(\mathsf{P} \xmapsto{\mathsf{size}} \mathsf{S}) \;*\; 0 < \mathsf{S} < 256 \end{array}\right\}$$

```
#define replace_dv(M, P, S) {\
  size_t DVS = M->dvsize;\
  if (DVS != 0) {\
    mchunkptr DV = M->dv;\
    assert(is_small(DVS));\
    insert_small_chunk(M, DV, DVS);\
  }\
  M->dvsize = S;\
  M->dv = P;\
}
```

$$\left\{\begin{array}{l}\exists v, \{U_i \mid i \in [0,64)\}.\, \texttt{least\_addr} = 5 \;*\; \ast_{i=0}^{32}.\, treebin_i(U_{i+32}) \\ *\; arena(A_{\mathsf{a}} \uplus (\uplus_{i=0}^{64}.\, U_i)_{\mathsf{u}} \uplus v, t) \\ *\; \ast_{i=0}^{32}.\, smallbin_i(U_i) \;*\; victim(v) \end{array}\right\}$$

Chapter 4

dlmalloc

Specification:

$\left\{state(A)\right\}$
`dlmalloc(bytes)`
$\left\{state(A \uplus \{\texttt{ret} \mapsto \lceil \texttt{bytes} \rceil_{\texttt{w}}\}) \ * \ *_{i=0}^{\lceil \texttt{bytes}/\texttt{w} \rceil} . \texttt{ret} + i\texttt{w} \mapsto \_ \ * \ \frac{1}{2}(\texttt{ret} - 2\texttt{w} \overset{\texttt{size}}{\longmapsto} \_)\right\}$

4.1 PAGE 1

4.2 PAGE 2

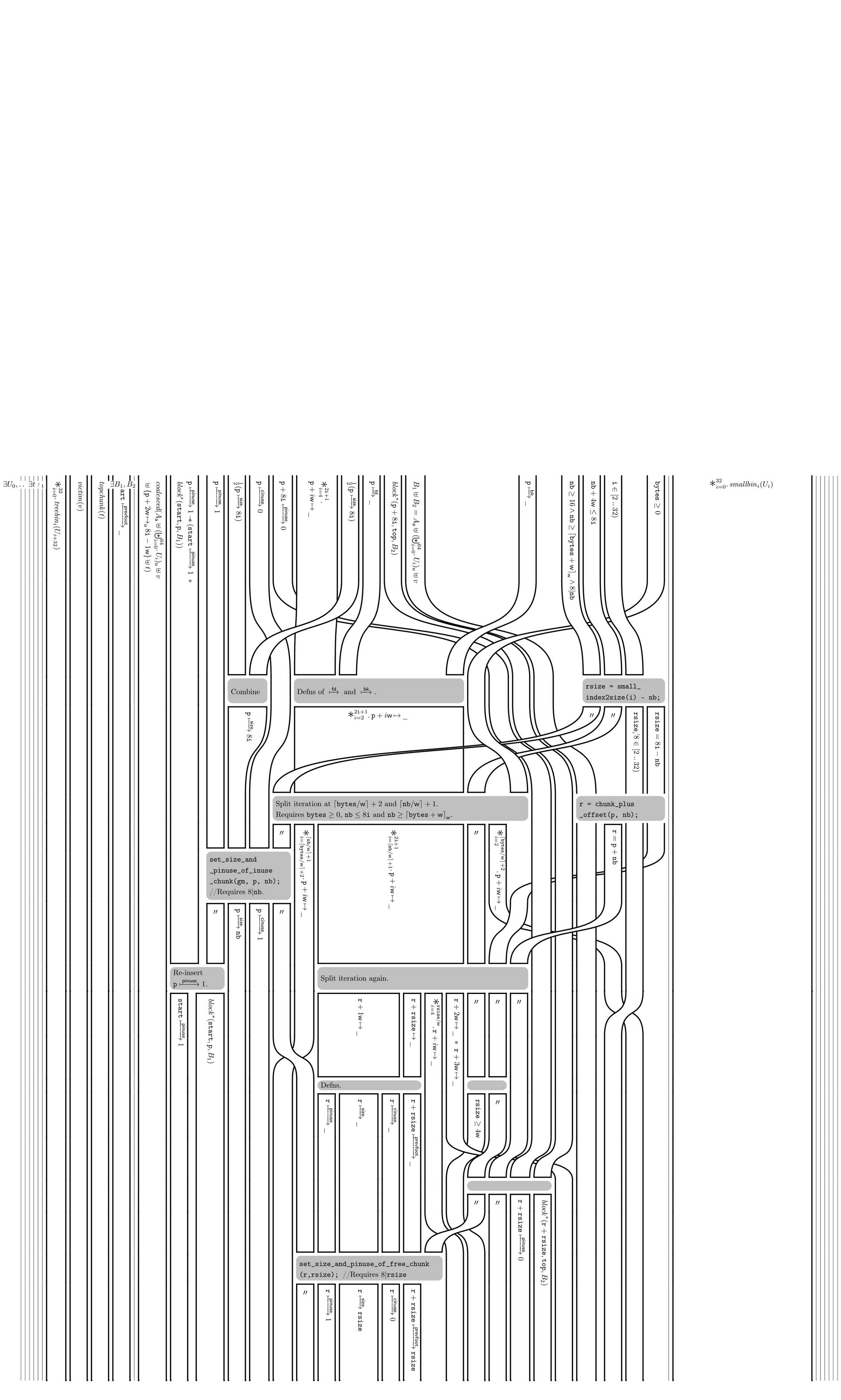
bytes ≥ 0
2 ≤ idx < 32
2idx ≥ ⌈bytes/w⌉ + 1
smallmap[idx] = 1
U_idx ≠ {}
(bmode | idx)⁺(p, b, U_idx)
Defn of (−)⁺ and bmode.
b = smallbin + 8idx
p ↦bk b
arena(A_a ⊎ (⊎_{i=0}^{64} U_i)_u ⊎ v, t)
topchunk(t)
victim(t)
*_{i=0}^{32} treebin_t(U_{i+32})
*_{i∈[0..32)⊎−idx} smallbin_t(U_i)
Since U_idx ⊎− {p + 2w ↦_u 8idx − 1w} is defined, we can replace U_idx with U_idx ⊎ {p + 2w ↦_u 8idx − 1w}.
unlink_first_small_chunk(gm, b, p, idx);
smallbin_idx(U_idx)
Recombine smallbins.
*_{i=0}^{32}. smallbin_i(U_i)
Defn of arena.
block^t(start, top, A_a ⊎ (⊎_{i=0}^{64}. U_i)_u ⊎ v ⊎ {p + 2w ↦_u 8idx − 1w})
Split block list.
∃B_1, B_2.
ublock(p, p + 8idx, {p + 2w ↦_u 8idx − 1w})
Extract p −pinuse→ 1.
Defn of ublock
Combine.
Defns of −fd→ and −bk→.
Split iteration at ⌈bytes/w⌉ + 2.
Requires bytes ≥ 0 and 8idx + w ≥ bytes.
Defn of −prevfoot→

4.3 PAGE 3

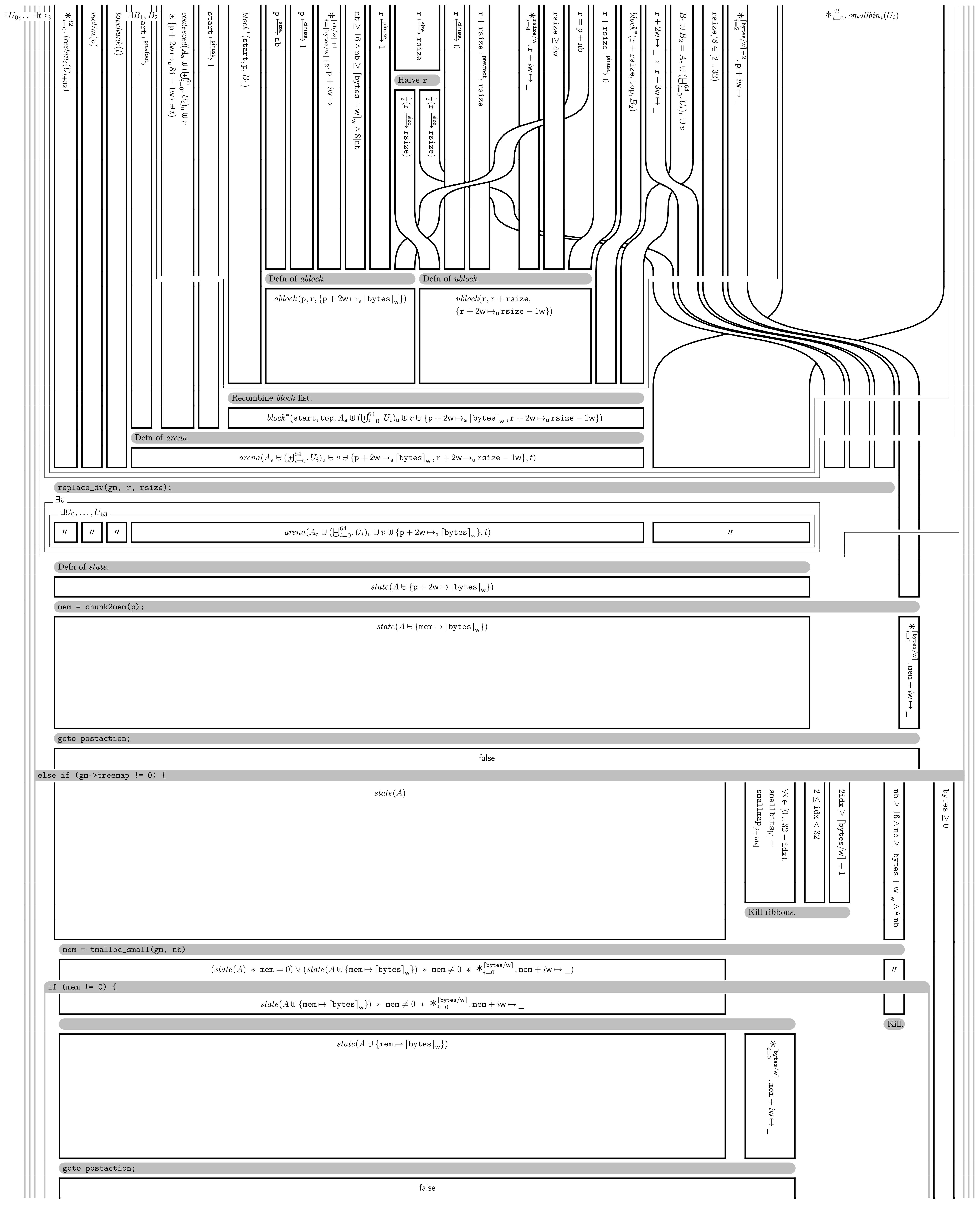
$*_{i=0}^{32}.\ smallbin_i(U_i)$
$2 \le \texttt{idx} < 32$
$B_1 \uplus B_2 = A_\texttt{a} \uplus (\biguplus_{i=0}^{64}.\ U_i)_\texttt{u} \uplus v$
$block^*(\texttt{p} + 8\texttt{idx}, \texttt{top}, B_2)$
$2\texttt{idx} \ge \lceil \texttt{bytes}/\texttt{w} \rceil + 1$
$*_{i=2}^{\lceil \texttt{bytes}/\texttt{w} \rceil + 2}.\ \texttt{p} + i\texttt{w} \mapsto \_$
$*_{i=\lceil \texttt{bytes}/\texttt{w} \rceil + 2}^{2\texttt{idx}+1}.\ \texttt{p} + i\texttt{w} \mapsto \_$
$\texttt{p} \xrightarrow{\texttt{cinuse}} 0$
$\texttt{p} + 8\texttt{idx} \xrightarrow{\texttt{pinuse}} 0$
$\texttt{p} \xrightarrow{\texttt{size}} \_$
$\texttt{p} \xrightarrow{\texttt{pinuse}} 1$
set_inuse_and_pinuse(gm, p, small_index2size(idx));
//
//
$\texttt{p} + 8\texttt{idx} \xrightarrow{\texttt{pinuse}} 1$
$\texttt{p} \xrightarrow{\texttt{cinuse}} 1$
Re-insert $\texttt{p} \xrightarrow{\texttt{pinuse}} 1$.
$\texttt{start} \xrightarrow{\texttt{pinuse}} 1$
$block^*(\texttt{start}, \texttt{p}, B_1)$
Defn of *ablock*. Requires $8\texttt{idx} \ge 4\texttt{w}$ and $\lceil \texttt{bytes} + \texttt{w} \rceil_\texttt{w} \le 8\texttt{idx}$.
$ablock(\texttt{p}, \texttt{p} + 8\texttt{idx}, \{\texttt{p} + 2\texttt{w} \mapsto_\texttt{a} \lceil \texttt{bytes} \rceil_\texttt{w}\})$
Recombine *block* list.
$block^*(\texttt{start}, \texttt{top}, A_\texttt{a} \uplus (\biguplus_{i=0}^{64}.\ U_i)_\texttt{u} \uplus v \uplus \{\texttt{p} + 2\texttt{w} \mapsto_\texttt{a} \lceil \texttt{bytes} \rceil_\texttt{w}\})$
Defn of *arena*.
$arena(A_\texttt{a} \uplus (\biguplus_{i=0}^{64}.\ U_i)_\texttt{u} \uplus v \uplus \{\texttt{p} + 2\texttt{w} \mapsto_\texttt{a} \lceil \texttt{bytes} \rceil_\texttt{w}\}, t)$
Defn of *state*.
$state(A \uplus \{\texttt{p} + 2\texttt{w} \mapsto \lceil \texttt{bytes} \rceil_\texttt{w}\})$
mem = chunk2mem(p);
$state(A \uplus \{\texttt{mem} \mapsto \lceil \texttt{bytes} \rceil_\texttt{w}\})$
$*_{i=0}^{\lceil \texttt{bytes}/\texttt{w} \rceil}.\ \texttt{mem} + i\texttt{w} \mapsto \_$
goto postaction;
false
} else if (nb > gm->dvsize) {
$state(A)$
$\texttt{bytes} \ge 0$
$\texttt{nb} \ge 16 \wedge \texttt{nb} \ge \lceil \texttt{bytes} + \texttt{w} \rceil_\texttt{w} \wedge 8 | \texttt{nb}$
$2\texttt{idx} \ge \lceil \texttt{bytes}/\texttt{w} \rceil + 1$
$2 \le \texttt{idx} < 32$
$\forall i \in [0\,..\,32 - \texttt{idx}).\ \texttt{smallbits}_{[i]} = \texttt{smallmap}_{[i+\texttt{idx}]}$
$\texttt{smallbits} \neq 0$
if (smallbits != 0) { /* Use chunk in next nonempty smallbin */
Defn of *state*.
$\exists t$
$\exists v$
$\exists U_0, \ldots, U_{63}$
$*_{i=0}^{32}.\ trebin_i(U_{i+32})$
$victim(v)$
$topchunk(t)$
$arena(A_\texttt{a} \uplus (\biguplus_{i=0}^{64}.\ U_i)_\texttt{u} \uplus v, t)$
$*_{i=0}^{32}.\ smallbin_i(U_i)$
binmap_t leftbits = (smallbits << idx) & left_bits(idx2bit(idx));
binmap_t leastbit = least_bit(leftbits);
compute_bit2idx(leastbit, i);
$\texttt{i} \in [2\,..\,32)$
$\texttt{smallmap}_{[\texttt{i}]} = 1$
$\texttt{nb} + 4\texttt{w} \le 8\texttt{i}$
Split off $\texttt{i}^{\text{th}}$ smallbin.
$*_{i \in [0..32) \uplus \texttt{i}.}\ smallbin_i(U_i)$
$smallbin_\texttt{i}(U_\texttt{i})$
b = smallbin_at(gm, i);
$\texttt{b} = \texttt{smallbin} + 8\texttt{i}$
$bin(|\texttt{i}|, \texttt{b}, U_\texttt{i})$
$\texttt{smallmap}_{[\texttt{i}]} = (U_\texttt{i} \neq \{\})$
//

4.4 PAGE 4

4.5 PAGE 5


bytes ≥ 0
i ∈ [2 .. 32)
nb + 4w ≤ 8i
nb ≥ 16 ∧ nb ≥ ⌈bytes + w⌉w ∧ 8|nb
p ↦bk _
B1 ⊎ B2 = Aa ⊎ (⊎i=0..64 Ui)u ⊎ v
block*(p + 8i, top, B2)
p ↦fd _
½(p ↦size 8i)
⊛i=4..2i+1 p + iw ↦ _
p + 8i ↦pinuse 0
p ↦cinuse 0
½(p ↦size 8i)
p ↦pinuse 1
p ↦pinuse 1 ⊛ (start ↦pinuse 1 ⊛ block*(start, p, B1))
coalesced(Aa ⊎ (⊎i=0..64 Ui)u ⊎ v ⊎ {p + 2w ↦u 8i − 1w} ⊎ t)
∃B1, B2
art ↦prevfoot _
topchunk(t)
victim(v)
⊛i=0..32 trebini(Ui+32)
∃U0, .. ∃t
⊛i=0..32 smallbini(Ui)
rsize = small_ index2size(i) - nb;
rsize = 8i − nb
rsize/8 ∈ [2 .. 32)
r = chunk_plus _offset(p, nb);
r = p + nb
Combine
Defns of ↦fd and ↦bk.
⊛i=2..2i+1 p + iw ↦ _
Split iteration at ⌈bytes/w⌉ + 2 and ⌈nb/w⌉ + 1.
Requires bytes ≥ 0, nb ≤ 8i and nb ≥ ⌈bytes + w⌉w.
p ↦size 8i
set_size_and _pinuse_of_inuse _chunk(gm, p, nb); //Requires 8|nb.
p ↦size nb
p ↦cinuse 1
⊛i=⌈nb/w⌉+1..2i+1 p + iw ↦ _
⊛i=2..⌈bytes/w⌉+2 p + iw ↦ _
Re-insert p ↦pinuse 1.
start ↦pinuse 1
block*(start, p, B1)
Split iteration again.
r + 1w ↦ _
r + rsize ↦ _
⊛i=4..rsize/w r + iw ↦ _
r + 2w ↦ _ * r + 3w ↦ _
rsize ≥ 4w
Defns.
r + rsize ↦prevfoot _
r ↦cinuse _
r ↦size _
r ↦pinuse _
set_size_and_pinuse_of_free_chunk (r,rsize); //Requires 8|rsize
r + rsize ↦prevfoot rsize
r ↦cinuse 0
r ↦size rsize
r ↦pinuse 1
block*(r + rsize, top, B2)
r + rsize ↦pinuse 0

4.6 PAGE 6



Halve r
Defn of ablock.
Defn of ublock.
ablock(p, r, {p + 2w ↦a ⌈bytes⌉w})
ublock(r, r + rsize, {r + 2w ↦u rsize − 1w})
Recombine block list.
block*(start, top, Aa ⊎ (⊎64 i=0. Ui)u ⊎ v ⊎ {p + 2w ↦a ⌈bytes⌉w , r + 2w ↦u rsize − 1w})
Defn of arena.
arena(Aa ⊎ (⊎64 i=0. Ui)u ⊎ v ⊎ {p + 2w ↦a ⌈bytes⌉w , r + 2w ↦u rsize − 1w}, t)
replace_dv(gm, r, rsize);
∃v
∃U0, . . . , U63
arena(Aa ⊎ (⊎64 i=0. Ui)u ⊎ v ⊎ {p + 2w ↦a ⌈bytes⌉w}, t)
Defn of state.
state(A ⊎ {p + 2w ↦ ⌈bytes⌉w})
mem = chunk2mem(p);
state(A ⊎ {mem ↦ ⌈bytes⌉w})
goto postaction;
false
else if (gm->treemap != 0) {
state(A)
Kill ribbons.
mem = tmalloc_small(gm, nb)
(state(A) ∗ mem = 0) ∨ (state(A ⊎ {mem ↦ ⌈bytes⌉w}) ∗ mem ≠ 0 ∗ ∗⌈bytes/w⌉ i=0 . mem + iw ↦ _)
if (mem != 0) {
state(A ⊎ {mem ↦ ⌈bytes⌉w}) ∗ mem ≠ 0 ∗ ∗⌈bytes/w⌉ i=0 . mem + iw ↦ _
Kill.
state(A ⊎ {mem ↦ ⌈bytes⌉w})
goto postaction;
false

4.7 PAGE 7

$state(A)$

$state(A)$

$nb \geq 16 \wedge nb \geq \lceil bytes + w \rceil_w \wedge 8 | nb$

$bytes \geq 0$

$state(A)$

$nb \geq 16 \wedge nb \geq \lceil bytes + w \rceil_w \wedge 8 | nb$

$bytes \geq 0$

```
else if (bytes >= MAX_REQUEST)
```

$state(A)$

$bytes \geq 0$

```
nb = MAX_SIZE_T;
```

$nb = 2^{32} - 8$

```
else {
```

$bytes \geq 256$

```
nb = pad_request(bytes);
```

$nb \geq 16 \wedge nb \geq \lceil bytes + w \rceil_w \wedge 8 | nb$

$bytes \geq 0$

```
if (gm->treemap != 0) {
mem = tmalloc_small(gm, nb)
```

$(state(A) \ * \ mem = 0) \vee (state(A \uplus \{mem \mapsto \lceil bytes \rceil_w\}) \ * \ mem \neq 0 \ * \ \ast_{i=0}^{\lceil bytes/w \rceil} .mem + iw \mapsto \_)$

```
if (mem != 0) {
```

$state(A \uplus \{mem \mapsto \lceil bytes \rceil_w\}) \ * \ mem \neq 0 \ * \ \ast_{i=0}^{\lceil bytes/w \rceil} .mem + iw \mapsto \_$

$state(A \uplus \{mem \mapsto \lceil bytes \rceil_w\})$

$\ast_{i=0}^{\lceil bytes/w \rceil} .mem + iw \mapsto \_$

```
goto postaction;
```

$state(A)$

```
}
}
```

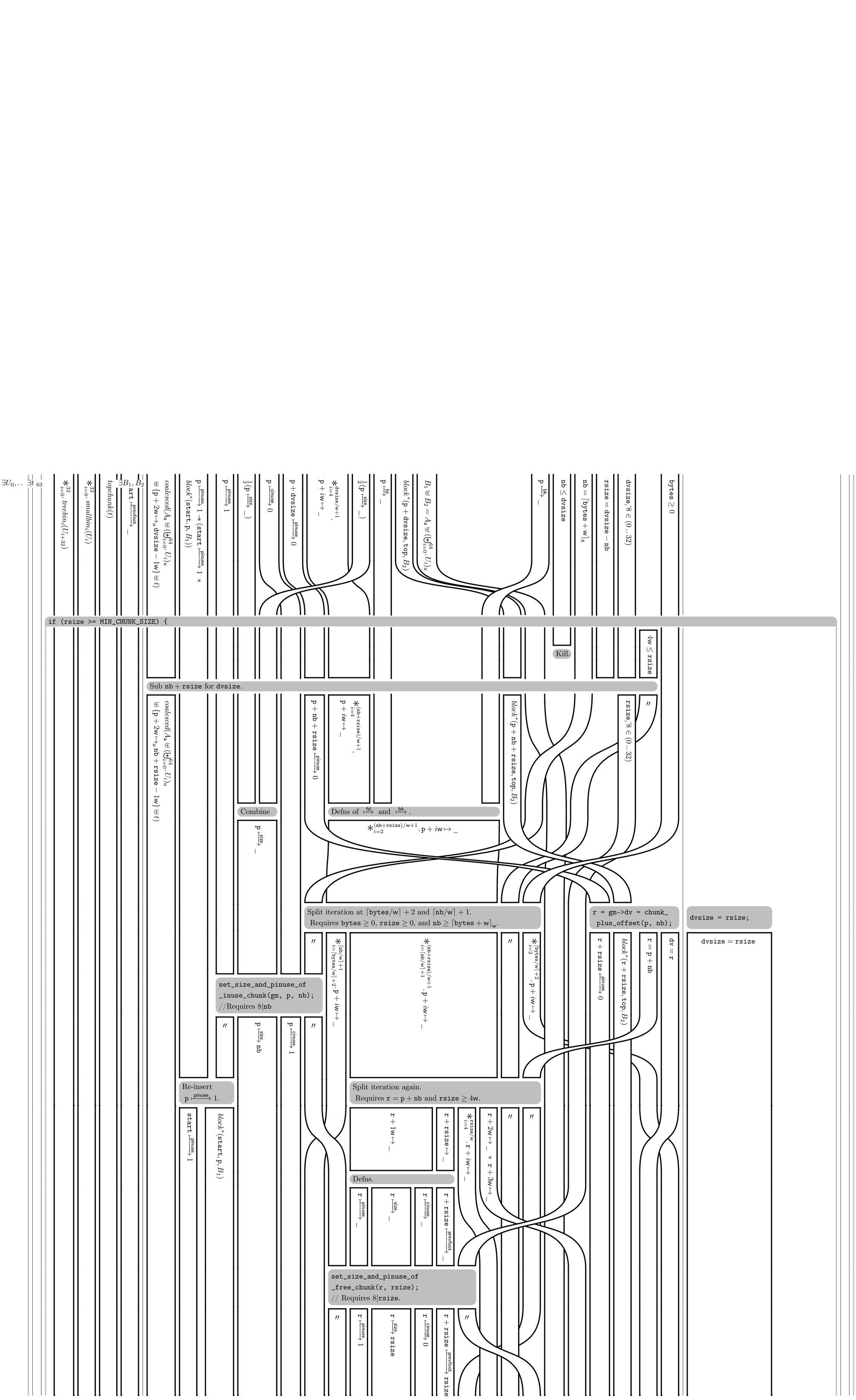
$nb \geq 16 \wedge nb \geq \lceil bytes + w \rceil_w \wedge 8 | nb$

```
}
```

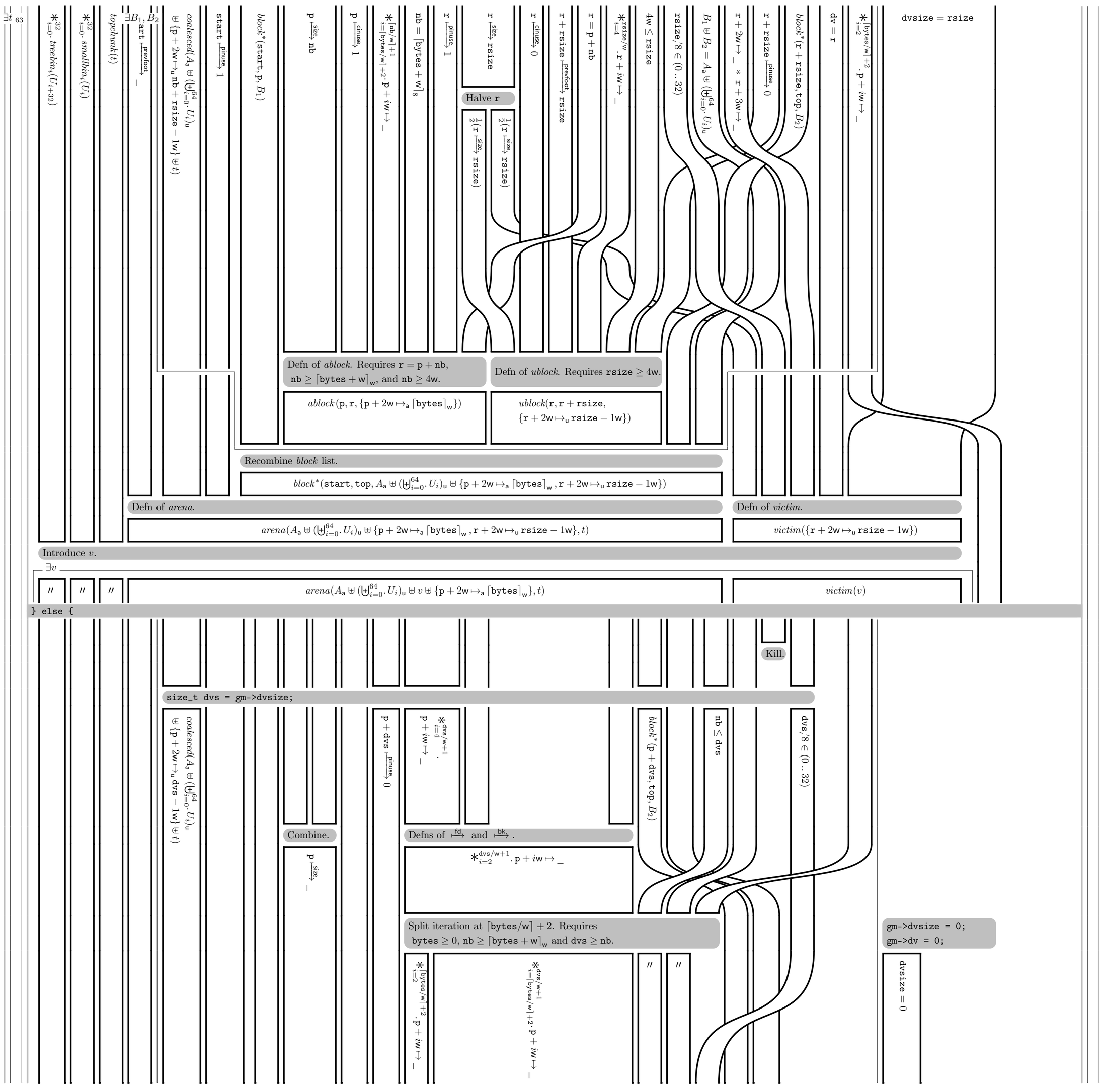
$nb = 2^{32} - 8 \vee (nb \geq 16 \wedge nb \geq \lceil bytes + w \rceil_w \wedge 8 | nb)$

4.8 PAGE 8

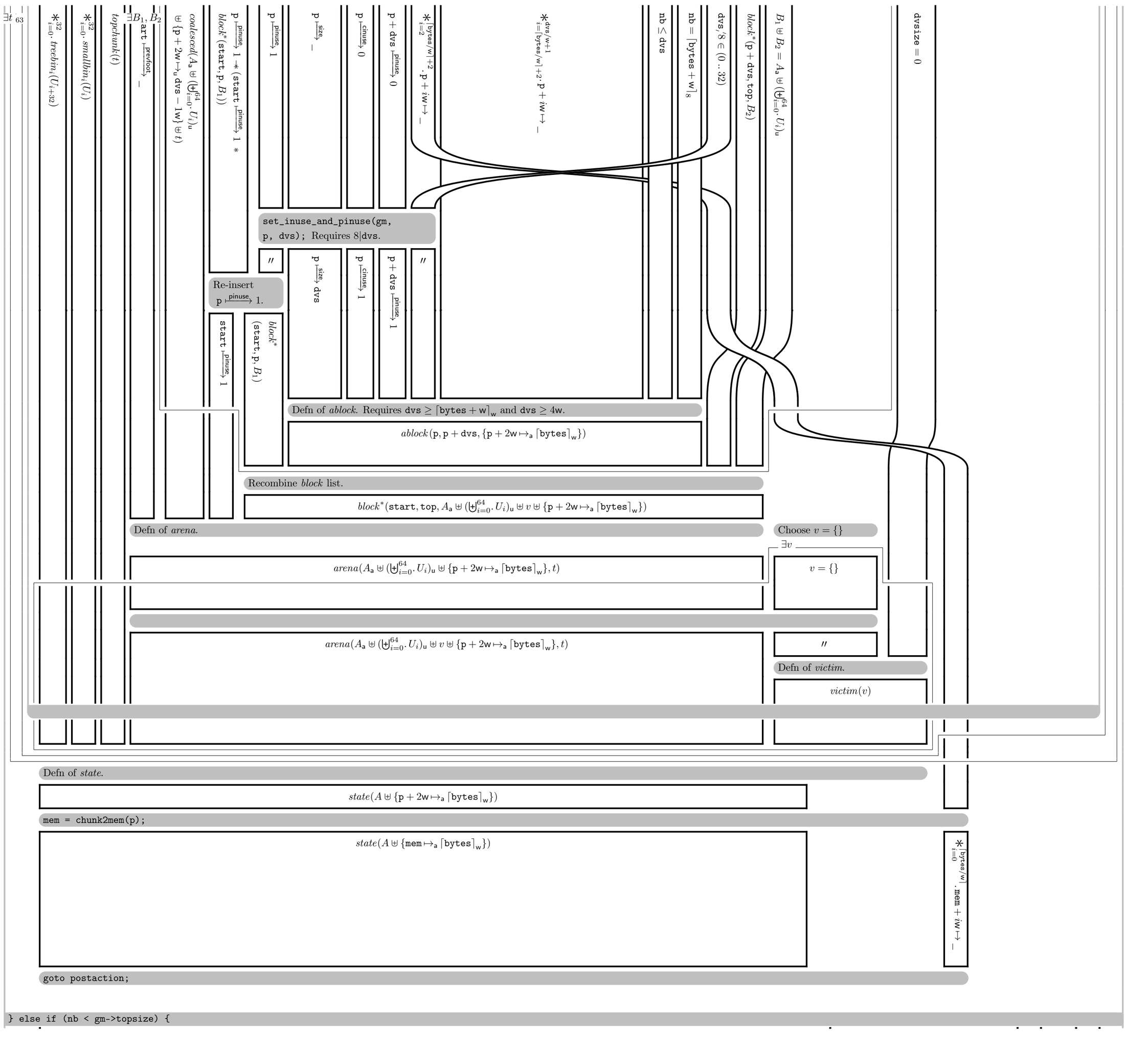
4.9 PAGE 9


bytes ≥ 0
dvsize/8 ∈ (0 .. 32)
rsize = dvsize − nb
nb = ⌈bytes + w⌉_s
nb ≤ dvsize
p ↦^bk _
B1 ⊎ B2 = Aa ⊎ (⊎_{i=0}^{64}. Ui)u
block*(p + dvsize.top, B2)
p ↦^fd _
½(p ↦^size _)
⊛_{i=4}^{dvsize/w+1}. p + iw ↦ _
p + dvsize ↦^pinuse 0
p ↦^cinuse 0
½(p ↦^size _)
p ↦^pinuse 1
p ↦^pinuse 1 −∗ (start ↦^pinuse 1 ∗ block*(start, p. B1))
coalesced(Aa ⊎ (⊎_{i=0}^{64}. Ui)u ⊎ {p + 2w ↦u dvsize − 1w} ⊎ t)
∃B1, B2
art ↦^prevfoot _
topchunk(t)
⊛_{i=0}^{32}. smallbini(Ui)
⊛_{i=0}^{32}. treebini(Ui+32)
∃U0, .. ∃t63 _
if (rsize >= MIN_CHUNK_SIZE) {
4w ≤ rsize
Kill.
Sub nb + rsize for dvsize.
dvsize/8 ∈ (0 .. 32)
rsize = dvsize − nb
nb = ⌈bytes + w⌉_s
nb ≤ dvsize
p ↦^bk _
B1 ⊎ B2 = Aa ⊎ (⊎_{i=0}^{64}. Ui)u
block*(p + dvsize.top, B2)
p ↦^fd _
½(p ↦^size _)
⊛_{i=4}^{dvsize/w+1}. p + iw ↦ _
p + dvsize ↦^pinuse 0
p ↦^cinuse 0
½(p ↦^size _)
p ↦^pinuse 1
p ↦^pinuse 1 −∗ (start ↦^pinuse 1 ∗ block*(start, p. B1))
coalesced(Aa ⊎ (⊎_{i=0}^{64}. Ui)u ⊎ {p + 2w ↦u dvsize − 1w} ⊎ t)
∃B1, B2
art ↦^prevfoot _
topchunk(t)
⊛_{i=0}^{32}. smallbini(Ui)
⊛_{i=0}^{32}. treebini(Ui+32)
Sub nb + rsize for dvsize.
coalesced(Aa ⊎ (⊎_{i=0}^{64}. Ui)u ⊎ {p + 2w ↦u nb + rsize − 1w} ⊎ t)
p + nb + rsize ↦^pinuse 0
⊛_{i=4}^{(nb+rsize)/w+1}. p + iw ↦ _
block*(p + nb + rsize, top, B2)
rsize/8 ∈ (0 .. 32)
″
Combine
p ↦^size _
Defns of ↦^fd and ↦^bk.
⊛_{i=2}^{(nb+rsize)/w+1}. p + iw ↦ _
Split iteration at ⌈bytes/w⌉ + 2 and ⌈nb/w⌉ + 1.
Requires bytes ≥ 0, rsize ≥ 0, and nb ≥ ⌈bytes + w⌉_w.
″
⊛_{i=⌈nb/w⌉+1}^{⌈bytes/w⌉+2}. p + iw ↦ _
⊛_{i=⌈nb/w⌉+1}^{(nb+rsize)/w+1}. p + iw ↦ _
″
⊛_{i=2}^{⌈bytes/w⌉+2}. p + iw ↦ _
r = gm->dv = chunk_ plus_offset(p, nb);
r + rsize ↦^pinuse 0
block*(r + rsize, top, B2)
r = p + nb
dv = r
dvsize = rsize;
dvsize = rsize
set_size_and_pinuse_of _inuse_chunk(gm, p, nb); //Requires 8|nb
″
p ↦^size nb
p ↦^cinuse 1
″
Re-insert p ↦^pinuse 1.
start ↦^pinuse 1
block*(start, p, B1)
Split iteration again.
Requires r = p + nb and rsize ≥ 4w.
r + rsize ↦ _
r + 1w ↦ _
⊛_{i=4}^{rsize/w}. r + iw ↦ _
r + 2w ↦ _ ∗ r + 3w ↦ _
″
″
Defns.
r + rsize ↦^prevfoot _
r ↦^cinuse _
r ↦^size _
r ↦^pinuse _
set_size_and_pinuse_of _free_chunk(r, rsize); // Requires 8|rsize.
″
r + rsize ↦^prevfoot rsize
r ↦^cinuse 0
r ↦^size rsize
r ↦^pinuse 1

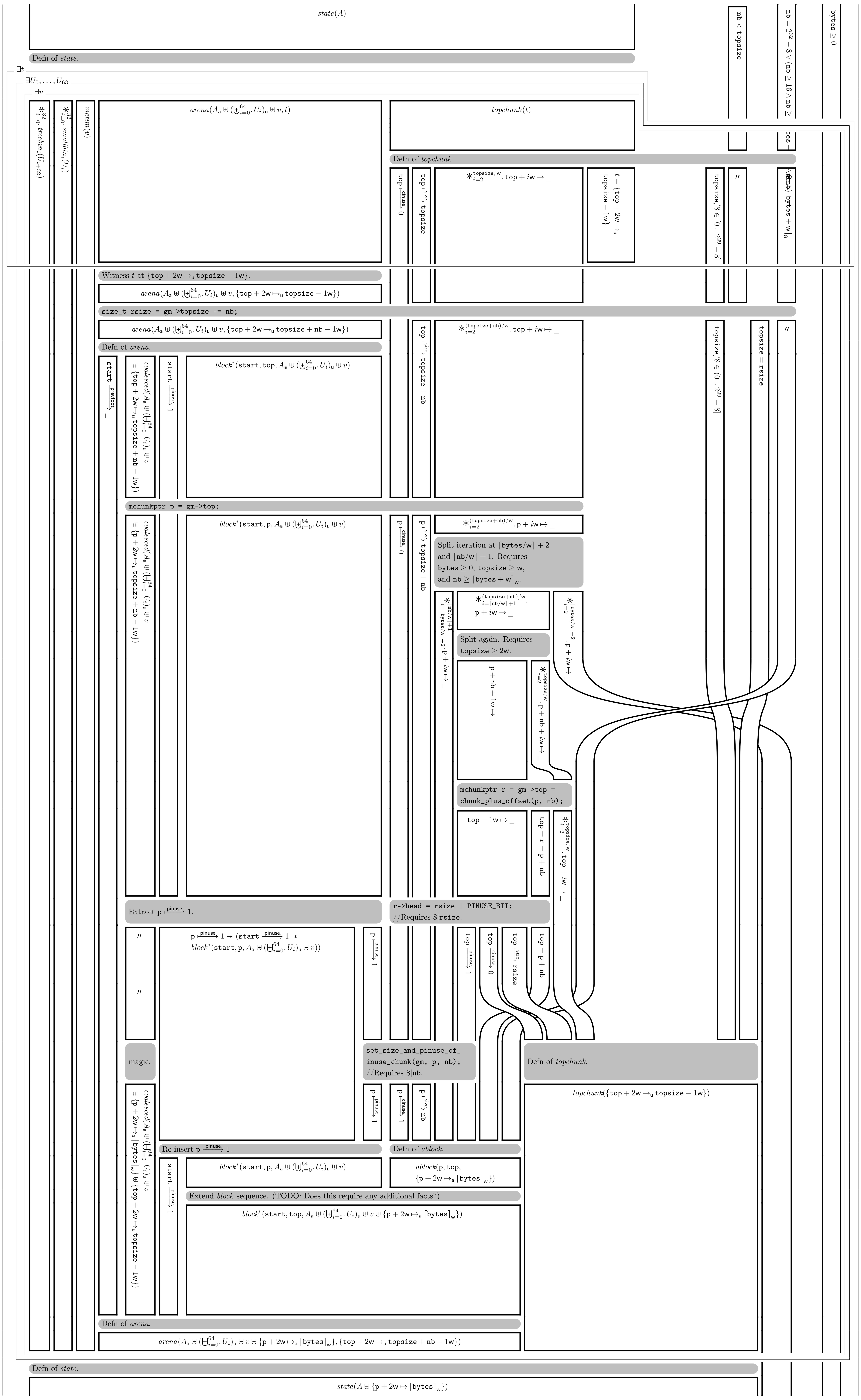
4.10 PAGE 10



4.11 Page 11



4.12 Page 12



state(A)
Defn of state.
∃t
∃U0, …, U63
∃v
arena(Aa ⊎ (⊎64 i=0. Ui)u ⊎ v, t)
topchunk(t)
Defn of topchunk.
nb < topsize
bytes ≥ 0
Witness t at {top + 2w ↦u topsize − 1w}.
arena(Aa ⊎ (⊎64 i=0. Ui)u ⊎ v, {top + 2w ↦u topsize − 1w})
size_t rsize = gm->topsize -= nb;
arena(Aa ⊎ (⊎64 i=0. Ui)u ⊎ v, {top + 2w ↦u topsize + nb − 1w})
Defn of arena.
topsize = rsize
block*(start, top, Aa ⊎ (⊎64 i=0. Ui)u ⊎ v)
mchunkptr p = gm->top;
block*(start, p, Aa ⊎ (⊎64 i=0. Ui)u ⊎ v)
Split iteration at ⌈bytes/w⌉ + 2 and ⌈nb/w⌉ + 1. Requires bytes ≥ 0, topsize ≥ w, and nb ≥ ⌈bytes + w⌉w.
Split again. Requires topsize ≥ 2w.
mchunkptr r = gm->top = chunk_plus_offset(p, nb);
top + 1w ↦ _
top = r = p + nb
Extract p ↦pinuse 1.
r->head = rsize | PINUSE_BIT; //Requires 8|rsize.
magic.
set_size_and_pinuse_of_ inuse_chunk(gm, p, nb); //Requires 8|nb.
Defn of topchunk.
topchunk({top + 2w ↦u topsize − 1w})
Re-insert p ↦pinuse 1.
Defn of ablock.
block*(start, p, Aa ⊎ (⊎64 i=0. Ui)u ⊎ v)
ablock(p, top, {p + 2w ↦a ⌈bytes⌉w})
Extend block sequence. (TODO: Does this require any additional facts?)
block*(start, top, Aa ⊎ (⊎64 i=0. Ui)u ⊎ v ⊎ {p + 2w ↦a ⌈bytes⌉w})
Defn of arena.
arena(Aa ⊎ (⊎64 i=0. Ui)u ⊎ v ⊎ {p + 2w ↦a ⌈bytes⌉w}, {top + 2w ↦u topsize + nb − 1w})
Defn of state.
state(A ⊎ {p + 2w ↦ ⌈bytes⌉w})