1 General Info

array(a; [x$_1$; x$_2$; : : : x$_n$]) , a 7! x$_1$ (a + 4) a 7! x$_2$: : : (a + 4 n) 7! x$_n$

For stacks:

a ; nil , a 7! NULL

a ; [x$_1$: : : ; x$_n$] , a 7! (x$_1$; a$_1$) : : : a$_{n\ 1}$ 7! (x$_n$; a$_n$) a$_n$ 7! NULL

For queues:

(hd; tl) ; nil , hd 7! NULL tl 7! NULL

(hd; tl) ; [x$_n$: : : ; x$_1$] , hd 7! (x$_1$; a$_1$) : : : a$_{n\ 1}$ 7! tl = (NULL ; x$_n$)

l#! R , l !$_0$ R and R is constant (i.e. doesn't depend on its variable)

1.1 Join Spawn rules

$$\frac{\text{f P gf f Qg} \quad \text{l fresh in F}}{\text{f F} \quad \text{P gSpawn f f F} \quad \text{l\#!} \quad \text{Qg}}\ \text{spwn}$$

$$\frac{}{\text{f l\#!} \quad \text{QgJoin(l) f Qg}}\ \text{join}$$

1.2 Histories

Hist , N 9 list list

t ,!$_h$ (l$_1$; l$_2$) , h[t] = (l$_1$; l$_2$)

bounded h , 9t: 8t^0 > t; t ^{0} 62h

last h , minf tj8t^0 > t; t ^{0} 62hg

listof h , $_2$(h[last h]) (i.e. (last h) ,! (_; listof h))

continuous h , 8t:t 2 h ^ (t + 1) 2 h ! 9 l:t ,! (_; l) ^ (t + 1) ,! (l; _)

gaplessh , 8t 2 h ! 8 t^0 < t; t ^{0} 2 h

init h , 0 ,! (; _)

stacklike h , 8t 2 h ! 9 l; x; t ,! (x :: l; l) _ t ,! (l; x :: l)

queuelike h , 8t 2 h ! 9 l; x; t ,! (l; x :: l) _ t ,! (l :: x; l)

stack_history h , continuous h^ gaplessh^ bounded v^ init h^ stacklike h

queue_history h , continuous h^ gaplessh^ bounded v^ init h^ queuelike h

2 Multiple-thread counter.

```
int main() f
  f empg
  a = malloc (n);
  f $\text{array}(a; [\_; \_; : : : ; \_]_n$ g
  (l, c) = malloc (LOCK_SIZE);
  f l 7! _ c 7! _ $\text{array}(a; [\_; \_; : : : ; \_]_n$ g
  c = 0;
  f l 7! _ c 7! 0 $\text{array}(a; [\_; \_; : : : ; \_]_n$ g
  MakeLock(l);
  f $l\,!^{1}_{0}$ R c 7! 0 Hold l; R; 0 $\text{array}(a; [\_; \_; : : : ; \_]_n$ g // R = v:c 7! v
  Release(l);
  f $l\,!^{1}_{0}$ R $\text{array}(a; [\_; \_; : : : ; \_]_n$ g
  f $F^{n\ 1}_{j=0}\ l\,!^{\frac{1}{n}}_{0}$ R $\text{array}(a; [\_; \_; : : : ; \_]_n$ g
  for (i = 0; i < n; i ++) f
  f $F^{i}_{j=0}\ l_j\,\#!\ R_j\ \ F^{n\ 1}_{j=i+1}\ l\,!^{\frac{1}{n}}_{0}$ R $\text{array}(a; [l_1; : : : ; l_i; \_; : : : \_]_n$ g
    a[i] = Spawn(incr, (l,c));
  f $F^{i+1}_{j=0}\ l_j\,\#!\ R_j\ \ F^{n\ 1}_{j=i+2}\ l\,!^{\frac{1}{n}}_{0}$ R $\text{array}(a; [l_1; : : : ; l_i; l_{i+1}; \_; : : : \_]_n$ g
  g
  f $F^{n}_{j=0}\ l_j\,\#!\ R_j$ $\text{array}(a; [l_1; : : : ; l_n]$g
  for (i = 0; i < n; i ++)
  f
  f $F^{i}_{j=0}\ R_j\ \ F^{n}_{j=i}\ l_j\,\#!\ R_j$ $\text{array}(a; [l_1; : : : ; l_n]$g
    Join(a[i]);
  f $F^{i+1}_{j=0}\ R_j\ \ F^{n}_{j=i+1}\ l_j\,\#!\ R_j$ $\text{array}(a; [l_1; : : : ; l_n]$g
  g
  f $F^{n}_{j=0}\ R_j$ $\text{array}(a; [l_1; : : : ; l_n]$g // $R_j = l\,!^{\frac{1}{n}}_{1}$ R
  f $l\,!_{n}$ R $\text{array}(a; [l_1; : : : ; l_n]$g
  free(a);
  f $l\,!_{n}$ Rg
  Acquire(l);
  f $l\,!_{n}$ R 9 $v_o$; c 7! (n + $v_o$) Hold l; R; (n + $v_o$)g
  f $l\,!_{n}$ R c 7! n Hold l; R; n g
  ret = c;
  f ret 7! n $l\,!_{n}$ R c 7! n Hold l; R; n g
  FreeLock (l);
  f ret 7! n l 7! 0 c 7! ng
```

```
   free(l,c);
   f ret 7! ng
   return   ret  g


void  incr(l,c)  f
   f l !^{1/n}_0  R g
   Acquire(l);
   f 9 v_o; c 7! v_o   Hold l; R; v_o   l !^{1/n}_0  Rg
   (    c)++;
   f 9 v_o; c 7! (v_o + 1)   Hold l; R; v_o   l !^{1/n}_0  Rg
   Release(l);
   f l !^{1/n}_1  Rg
g
```

3 Single Initialize / concurrent read

```
f l !_? R g     nn R = v:init 7! 0 ^ v = ? _ init 7! 1 d ^>7!^v data ^ [> > v]
data first_access(l) f
  f l !_? R g
  Acquire(l);
  f 9 v_o; init 7! 0 ^ v_o = ? _ init 7! 1 d 7!^{s_o} data  Hold l; R; v_o  l !_? R g
 nn where s_o = >  v_o
  if (init) f
    f init 7! 1 d 7!^{s_o} data  Hold l; R; v_o  l !_? R g
    f d 7!^{s_o/2} data  init 7! 1 d ^>  7!^{(v_o + s_o/2)} data  Hold l; R; v_o  l !_? R g
    Release(l);
    f d 7!^{s_o/2} data  l !_{s_o/2} R g
    return d;
    f d 7!^{s_o/2} data  l !_{s_o/2} R ^ ret = d g
  g
  else f
    f init 7! 0  Hold l; R; ?  l !_? R g
    InitializeData (d);
    f d 7! data  init 7! 0  Hold l; R; ?  l !_? R g
    init = 1;
    f d 7! data  init 7! 1  Hold l; R; ?  l !_? R g
    f d 7!^{1/2} data  d 7!^{1/2} data  init 7! 1  Hold l; R; ?  l !_? R g
    Release(l)
    f d 7!^{1/2} data  l !_? R g
    return d;
    f d 7!^{1/2} data  l !_? R ^ ret = d g
  g
g
f 9 _s; d 7!^s data  l !_? R ^ ret = d g
```

4 Stack Producer/consumer

```
f emp g
void create();
f hd ;   g


f hd ;   g
void delete();
f emp g

f hd ; ls g
void isemp();
f hd ; ls ^
    ls =  ^ ret = true _
    9x; l 0:l = x :: l ^ ret = false g

f hd ; ls g
void enq(int x);
f hd ; x :: ls g

f hd ; x :: ls g
void deq();
f hd ; ls ^ ret = x g

/  Producer  /
f l !_? R g   nn R = h: hd ; (listof( h)) ^ history_stack h
void produce(x, l) f
  f l !_? R g
  Acquire(l);
  f 9 h_o; hd ; l ^ history_stack h  Hold l; R; h_o  l !_? R g   nn l = listof( h_o)
  enq(x);
  f hd ; x :: l ^ history_stack h  Hold l; R; h_o  l !_? R g
  f  hd ; (listof( h_o + t ,!  (l; x :: l))) ^ history_stack (h_o + t ,!  (l; x :: l))
      Hold l; R; h_o  l !_? R g   nn t = last h_o + 1
  Release(l);
  f l !_{t,! (l;x ::l)} R g
g f l !_{t,! (l;x ::l)} R g

/  Consumer  /
f l !_? R g   nn R = h: hd ; (listof( h)) ^ history_stack h
void consume(l)f
  f l !_? R g
  bool cont = true;
  f cont = true ^ l !_? R g
  while (cont) f
    Acquire(l);
    f cont = true ^ 9 h_o; hd ; l ^ history_stack h
      Hold l; R; h_o l !_? R g nnl = listof( h_o)
    if (isemp() ) f
      Release(l);
      f cont = true ^ l !_? R g
    g else f
      f 9 x; l^0:l = x :: l ^ cont = true ^
     hd ; l ^ history_stack h Hold l; R; h_o l !_? R g
      ret = deq();
      f ret = x ^ cont = true ^
     hd ; l^0 ^ history_stack h Hold l; R; h_o l !_? R g
      f ret = x ^ cont = true ^
      hd ; (listof( h_o + t ,! (l; l^0))) ^ history_stack h
      Hold l; R; h_o l !_? R g     nn t = last h_o + 1
      Release(l);
      f ret = x ^ cont = true ^ l !_{t,! (l;l^0)} R g
      cont = false;
      f ret = x ^ cont = false ^ l !_{t,! (l;l^0)} R g
    g
    f cont = true ^ l !_? R _ cont = false ^ ret = x ^ l !_{t,! (l;l^0)} R g
  g
  f cont = false ^ ret = x ^ l !_{t,! (l;l^0)} R g
  return ret;
g
f ret = x ^ l !_{t,! (x :: l^0;l^0)} R g
```

```
/    Organizer    /
f l !    R   a 7! _   b 7! _ g     nn   R =   h:   hd ;    (listof( h)) ^ history _stack h
     ?
void   organize1(l ,  a ,  b)f
   f l !    R   a 7! _   b 7! _ g
        ?
   (t1 ,  v1)  =  consume(l);
   f (t1; v1) = x ^ l  !                 R   a 7! _   b 7! _ g
                        t1,!  (x :: l;l )
   if  (t1)  f
      f t1 7! 1  (t; v1) = x ^ l  !                 R   a 7! _   b 7! _ g
                                   t,!  (x :: l;l )
      a  =  v1;
      f t1 7! 1  (t; v1) = x ^ l  !                 R   a 7! v1   b 7! _ g
                                   t,!  (x :: l;l )
   g  else  f
      f t1 7! 0  (t; v1) = x ^ l  !                 R   a 7! _   b 7! _ g
                                   t,!  (x :: l;l )
      b  =  v1;
      f t1 7! 0  (t; v1) = x ^ l  !                 R   a 7! _   b 7! v1 g
                                   t,!  (x :: l;l )
   g
g
f (t1; v1) = x ^ l  !                 R
                     t,!  (x :: l;l )
      t1 7! 1   a 7! v1   b 7! _ _
      t1 7! 0   a 7! _   b 7! v1 g

f l !    R   a 7! _   b 7! _ g
     ?
void   organize2(l ,a ,b)f
   f l !    R   a 7! _   b 7! _ g
        ?
   organize1(l ,a ,b);
   f (t1; v1) = x ^ l  !                 R
                        t,!  (x :: l;l )
      t1 7! 1   a 7! v1   b 7! _ _
      t1 7! 0   a 7! _   b 7! v1 g
   organize1(l ,a ,b);
g
f (t2; v2) = x0^ (t1; v1) = x ^ l    !                                    R
                                      t,!  (x :: l;l )  t0,!  (x0:: l0;l0)
      t1 7! 1   t2 7! 1   a 7! v2   b 7! _ _
      t1 7! 1   t2 7! 0   a 7! v1   b 7! v2 _
      t1 7! 0   t2 7! 1   a 7! v2   b 7! v1 _
      t1 7! 0   t2 7! 0   a 7! _   b 7! _ g
```

```
void main() f
  f emp g
  l, x, y,z,a,b, = malloc (LOCK, LOCK, LOCK, LOCK,    int , int , );
  hd = create();
   f hd ;     l 7! _  a 7! _  b 7! _ g
  MakeLock(l);  nn  R =  h:  hd ;  (listof( h)) ^ history_stack h
  f hd ;     l ! $^{>}_{emp}$  R  Hold l; R; emp  a 7! _  b 7! _ g
  Release(l);
  f l ! $^{>}_{emp}$  R  a 7! _  b 7! _ g
  x = Spawn(produce, ((0,0), l));
  f x#!  $R_x$  l ! $^{\frac{2}{3}}_{emp}$  R  a 7! _  b 7! _ g  nn $R_x$ = l  ! $^{\frac{1}{3}}_{t,!\ (l; (0;0)):: l)}$  R
  y = Spawn(produce, ((1,1), l));
  f y#!  $R_y$  x#!  $R_x$  l ! $^{\frac{1}{3}}_{emp}$  R  a 7! _  b 7! _ g  nn $R_y$ = l  ! $^{\frac{1}{3}}_{t,!\ (l; (1;1)):: l)}$  R
  z = Spawn(organize2, (l, a, b));
  f z#!  $R_z$  y#!  $R_y$  x#!  $R_x$ g
  Join(x);
  f $h_1$ = $t_x$ ,!  ($l_x$; (0; 0) :: $l_x$) ^ z#!  $R_z$  y#!  $R_y$  l  ! $^{\frac{1}{3}}_{h_1}$  R g
  Join(y);
  f $h_2$ = $t_x$ ,!  ($l_x$; (0; 0) :: $l_x$)  $t_y$ ,!  ($l_y$; (1; 1) :: $l_y$) ^ z#!  $R_z$  l  ! $^{\frac{2}{3}}_{h_2}$  R g
  Join(z);
  f $h_3$ = $h_2$  $t_z$1 ,!  (x :: $l_z$1; $l_z$1)  $t_z$2 ,!  (y :: $l_z$2; $l_z$2)) ^ (k1; v1) = x ^ (k2; v2) = y ^
    l  ! $^{>}_{h_3}$  R
    k1 7! 1  k2 7! 1  a 7! v2  b 7! _ _
    k1 7! 1  k2 7! 0  a 7! v1  b 7! v2 _
    k1 7! 0  k2 7! 1  a 7! v2  b 7! v1 _
    k1 7! 0  k2 7! 0  a 7! _  b 7! _ g
  Acquire(l);
  f (k1; v1) = x ^ (k2; v2) = y ^
    l  ! $^{>}_{h_3}$  R  Hold l; R; h o  hd ;  (listof( $h_3$)) ^ history_stack ($h_3$)
    k1 7! 1  k2 7! 1  a 7! v2  b 7! _ _
    k1 7! 1  k2 7! 0  a 7! v1  b 7! v2 _
    k1 7! 0  k2 7! 1  a 7! v2  b 7! v1 _
    k1 7! 0  k2 7! 0  a 7! _  b 7! _ g     nn  Case analysis on h3
  f l  ! $^{>}_{h_3}$  R  Hold l; R; h o  hd ;  ( )
    (1; 1) = x ^ (0; 0) = y ^ k1 7! 1  k2 7! 0  a 7! 1  b 7! 0 _
    (0; 0) = x ^ (1; 1) = y ^ k1 7! 0  k2 7! 1  a 7! 1  b 7! 0 g
  Free(l);  free(k1,k2);
  f hd ;     a 7! 1  b 7! 0 g
  delete();
g
f a 7! 1  b 7! 0 g
```

5 Queue Producer/consumer

```
struct  elem f
  struct  elem  next;
  struct  elem  data;
g;

struct  fifo f
  struct  elem  hd;
  struct  elem  tl;
g;

f emp g
fifo  create()f
  Q = malloc( sizeof(fifo));
  f Q:hd 7! _  Q:hd 7! _ g
  hd, tl = NULL;
  f Q;    g
  return  Q;
g
f  Q;    g


f Q;    g
void  delete(Q)f
  free(Q);
g
f emp g

f Q;  ls g
void  isemp()f
  return  (Q.hd == NULL)
  f Q;  ls ^ ret = ( Q:hd ==  NULL ) g
g
f  Q;  ls ^
    ls =   ^ ret = true _
    9x;l0:l = l :: x ^ ret = false g

f Q;  ls g
void  enq(fifo Q, type x) f
  f Q;  ls g
  if (hd==NULL)   f
    f Q;    ^ hd = NULL  g
    Q! hd=(NULL, x);
    Q! tl=(NULL, x);
    f Q;  x ::   g
  g
    else f
    f Q;  [x1;:::;xn] ^ hd 6= NULL   g
```

```
      tl ! next = (NULL, x);
      f Q:hd 7! (x1;a1)  :::an 1 7! tl = ( an;xn)  an 7! (NULL ;x) g
      Q! tl=( nNULL, x);
      f Q:hd 7! (x1;a1)  :::an 1 7! (an;xn)  an 7! tl = (NULL ;x) g
      f Q;  [x;xn;:::;x1] g
    g
  g
f Q;  x :: ls g

f Q;  ls :: x g
void deq(fifo Q)f
   h=Q! hd! data;
   f h = x ^ Q;  ls :: x g
   n=Q! hd! next;
   f h = x ^ n = a1 ^ Q:hd 7! (x;a1)  a1 7! (x2;a2)  :::an 1 7! tl = (NULL ;xn) g
   Q! head=n;
   f h = x ^ n = a1 ^ Q:hd 7! (x2;a2)  :::an 1 7! tl = (NULL ;xn) g
   return h;
gf Q;  ls ^ ret = x g

/ Producer /
f l !?  R g   nn  R =  h:  Q;  (listof( h)) ^ history_queue h
void produce(fifo Q, type x, lock l) f
   f l !?  R g
   Acquire(l);
   f 9 ho; Q;  l ^ history_queue h  Hold l;R;ho  l !?  R g   nn  l = listof( ho)
   enq(x);
   f  Q;  x :: l ^ history_queue ho  Hold l;R;ho  l !?  R g
   f   Q;  (listof( ho  t,!  (l;x :: l))) ^ history_queue (ho  t,!  (l;x :: l))
        Hold l;R;ho  l !?  R g   nn  t = last ho + 1
   Release(l);
   f l  !t,!  (l;x ::l)  R g
g f l  !t,!  (l;x ::l)  R g

/ Consumer /
f l !  R g   nn  R =  h:  Q;  (listof( ho)) ^ history_queue h
void consume(l)f
   f l !  R g
   bool cont = true;
   f cont = true ^ l !  R g
   while (cont) f
      Acquire(l);
```

```
        f  cont =  true ^ 9 $h_o$;  Q ;    l ^ history _queue $h_o$
          Hold  l; R; $h_o$    l !      R  g   nnl = listof( $h_o$)
        if   (isemp()  )   f
           Release(l);
           f  cont =  true  ^ l !      R  g
        g  else   f
           f 9 x; $l^0$:l  =  l  :: x ^  cont =  true  ^
         Q ;    l ^  history _queue $h_o$    Hold  l; R; $h_o$    l !      R  g
           ret  =  deq();
           f  ret  =  x ^  cont =  true  ^
         Q ;    $l^0$ ^  history _queue $h_o$    Hold  l; R; $h_o$    l !      R  g
           f  ret  =  x ^  cont =  true  ^
          Q ;    (listof( $h_o$     t ,!    (l; $l^0$))) ^  history _queue $h_o$
          Hold  l; R; $h_o$    l !      R  g       nn   t = last  $h_o$ + 1
           Release(l);
           f  ret  =  x ^  cont =  true  ^ l   !$_{t,!\ \ (l;l^0)}$   R  g
           cont  =  false;
           f  ret  =  x ^  cont =  false  ^ l   !$_{t,!\ \ (l;l^0)}$   R  g
        g
        f  cont =  true  ^ l !      R _  cont =  false  ^  ret  =  x ^ l   !$_{t,!\ \ (l;l^0)}$   R  g
    g
    f  cont =  false  ^  ret  =  x ^ l   !$_{t,!\ \ (l;l^0)}$   R  g
    return    ret;
g
f  ret  =  x ^ l    !$_{t,!\ \ (l^0 :: x;l^0)}$   R  g
```

```
/  Organizer  /
f l !   R  a 7! _  b 7! _ g   nn  R =  h:  Q ;   (listof( h)) ^ history _queue ho
void  organize1(l, a, b)f
  f l !   R  a 7! _  b 7! _ g
  (t1, v1) = consume(l);
  f (t1; v1) = x ^ l  ! t1,!  (l :: x;l )   R  a 7! _  b 7! _ g
  if (t1) f
    f t1 7! 1  (t; v1) = x ^ l  ! t,!  (l :: x;l )   R  a 7! _  b 7! _ g
    a = v1;
    f t1 7! 1  (t; v1) = x ^ l  ! t,!  (l :: x;l )   R  a 7! v1  b 7! _ g
  g else f
    f t1 7! 0  (t; v1) = x ^ l  ! t,!  (l :: x;l )   R  a 7! _  b 7! _ g
    b = v1;
    f t1 7! 0  (t; v1) = x ^ l  ! t,!  (l :: x;l )   R  a 7! _  b 7! v1 g
  g
g
f (t1; v1) = x ^ l  ! t,!  (l :: x;l )   R
    t1 7! 1  a 7! v1  b 7! _ _
    t1 7! 0  a 7! _  b 7! v1 g

f l !   R  a 7! _  b 7! _ g
void  organize2(l,a,b)f
  f l !   R  a 7! _  b 7! _ g
  organize1(l,a,b);
  f (t1; v1) = x ^ l  ! t,!  (l :: x;l )   R
    t1 7! 1  a 7! v1  b 7! _ _
    t1 7! 0  a 7! _  b 7! v1 g
  organize1(l,a,b);
g
 f Rz = ( t2; v2) = x0 ^ (t1; v1) = x ^ l  ! t,!  (l :: x;l )  t0,!  (l0:: x0;l0)   R
    t1 7! 1  t2 7! 1  a 7! v2  b 7! _ _
    t1 7! 1  t2 7! 0  a 7! v1  b 7! v2 _
    t1 7! 0  t2 7! 1  a 7! v2  b 7! v1 _
    t1 7! 0  t2 7! 0  a 7! _  b 7! _ g
```

```
void  main() f
  f emp g
  l , x , y ,z ,a ,b , = malloc (LOCK, LOCK, LOCK, LOCK,    int , int , );
  hd = create();
  f Q ;    l 7! _ a 7! _ b 7! _ g
  MakeLock(l);  nn  R =  h:  Q ;  (listof( h)) ^ history_queue h
  f Q ;    l !_emp^>  R  Hold l; R; emp  a 7! _  b 7! _ g
  Release(l);
  f l !_emp^>  R  a 7! _  b 7! _ g
  x = Spawn(produce , ((0 ,0) , l );
  f x#!  R_x  l !_emp^(2/3)  R  a 7! _  b 7! _ g   nn R_x = l  !_(t,!  (l; (0 ;0)):: l))^(1/3)  R
  y = Spawn(produce , ((1 ,1) , l );
  f y#!  R_y  x#!  R_x  l !_emp^(1/3)  R  a 7! _  b 7! _ g   nn R_y = l  !_(t,!  (l; (1 ;1)):: l))^(1/3)  R
  z = Spawn(organize2 , (l , a , b));
  f z#!  R_z  y#!  R_y  x#!  R_x g
  Join(x);
  f h_1 = t_x ,!  (l_x ; (0; 0) :: l_x ) ^ z#!  R_z  y#!  R_y  l !_(h_1)^(1/3)  R g
  Join(y);
  f h_2 = t_x ,!  (l_x ; (0; 0) :: l_x )   t_y ,!  (l_y ; (1; 1) :: l_y ) ^ z#!  R_z  l !_(h_2)^(2/3)  R g
  Join(z);
  f h_3 = h_2  t_z1 ,!  (l_z1 :: x; l_z1)  t_z2 ,!  (l_z2 :: x^0; l_z2))^ (k1; v1) =  x^ (k2; v2) =  x^0^
    l  !_(h_3)^>  R
    k1 7! 1  k2 7! 1  a 7! v2  b 7! _ _
    k1 7! 1  k2 7! 0  a 7! v1  b 7! v2 _
    k1 7! 0  k2 7! 1  a 7! v2  b 7! v1 _
    k1 7! 0  k2 7! 0  a 7! _  b 7! _ g
  Acquire(l);
  f (k1; v1) =  x ^ (k2; v2) =  y ^
    l  !_(h_3)^>  R  Hold l; R; h_o   Q ;  (listof( h_3)) ^ history_queue (h_3)
    k1 7! 1  k2 7! 1  a 7! v2  b 7! _ _
    k1 7! 1  k2 7! 0  a 7! v1  b 7! v2 _
    k1 7! 0  k2 7! 1  a 7! v2  b 7! v1 _
    k1 7! 0  k2 7! 0  a 7! _  b 7! _ g    nn  Case analysis on h3
  f l  !_(h_3)^>  R  Hold l; R; h_o   Q ;  ( )
    (1; 1) =  x ^ (0; 0) =  y ^ k1 7! 1  k2 7! 0  a 7! 1  b 7! 0 _
    (0; 0) =  x ^ (1; 1) =  y ^ k1 7! 0  k2 7! 1  a 7! 1  b 7! 0 g
  Free(l);  free(k1 ,k2);
  f Q ;    a 7! 1  b 7! 0 g
  delete();
g
f a 7! 1  b 7! 0 g
```

6 Tree add

```
struct node
f
  int k;           //key _value
  struct node l; //left subtree
  struct node r; //right subtree
g;

void AddTree(struct node t, int res)f
  f t tree res 7! _ g
  if (empty(t)) f
    f t res 7! _ g
    res = 0;
    f t res 7! 0 g
  g else f
    f t (k; ltree; rtree ) res 7! _ g
    int lres, rres;
    thread lth, rth;
    f t (k; ltree; rtree ) (res; lres; rres; lth; rth ) 7! _ g
    f t 7! (k; l; r ) l ltree r rtree (res; lres; rres; lth; rth ) 7! _ g
    lthread = spawn (AddTree, (left, t ! l, lies));
    f lth #! Rl t 7! (k; l; r ) r rtree (res; rres; rth ) 7! _ g
    rthread = spawn (AddTree, (right, t ! r));
    f rth #! Rr lth #! Rl t 7! (k; l; r ) res 7! _ g
    join (lth);
    f (add_tree(ltree ) = kl ) ^ l ltree lres 7! kl )
    rth #! Rr t 7! (k; l; r ) res 7! _ g
    join (rth);
    f (add_tree(ltree ) = kl ) ^ l ltree lres 7! kl )
    (add_tree(rtree ) = kr ) ^ r rtree rres 7! kr )
    t 7! (k; l; r ) (res) 7! _ g
    res = lres + rres + t.k;
    f (add_tree(ltree ) = kl ) ^ lres 7! kl )
    (add_tree(rtree ) = kr ) ^ rres 7! kr )
    t (k; ltree; rtree ) (res) 7! (kl + kr + k) g
  g
  f (add_tree(tree)) = k0 ^ t tree (res) 7! (k0) g
g
```

7 Tree add with reporting

```
struct node
f
  lock l;
  int  k;          //sum _value
  int k;           //key _value
  struct node l; //left subtree
  struct node r; //right subtree
g;


f node:l !_? R g  //  R = v:STUFF
void AddTreeRep(struct node t, int RL)f
  f t  tree RL !_0 R g
  if (empty(t)) f
    nnThis branch is useless in practice.
    f add_tree( ) = 0 ^ t     RL !_0 R g
  g else f
    f t  (k; ltree; rtree )  RL !_0 R g
    f t 7! (k; l; r )  l  ltree  r  rtree  RL !_0 R g
    lthread = spawn (AddTreeRep, (left, t ! l, lies));
    f lth #! R_l  t 7! (k; l; r )  r  rtree  RL !^{2/3}_0 R g
    rthread = spawn (AddTreeRep, (right, t ! r));
    f rth #! R_r  lth #! R_l  t 7! (k; l; r )  RL !^{3}_0 R g
    Acquire(RL);
    f 9 v_o:result 7! (v_o + 0)  Hold RL; R; v_o
    rth #! R_r  lth #! R_l  t 7! (k; l; r )  RL !^{3}_0 R g
    result = result + (t ! k);
    f 9 v_o:result 7! (v_o + k)  Hold RL; R; v_o
    rth #! R_r  lth #! R_l  t 7! (k; l; r )  RL !^{3}_0 R g
    Release(RL);
    f rth #! R_r  lth #! R_l  t 7! (k; l; r )  RL !^{3}_k R g
    join (lth);
    f (add_tree(ltree) = k_l ^ l  ltree  RL !^{3}_{k_l} R)
    rth #! R_r  t 7! (k; l; r )  RL !^{3}_k R g
    join (rth);
```

f (add_tree(rtree) = k_r ^ r rtree RL $!^{\overline{3}}_{k_r}$ R)

(add_tree(ltree) = k_l ^ l ltree RL $!^{\overline{3}}_{k_l}$ R)

t 7! (k; l; r) RL $!^{\overline{3}}_{k}$ R g

f add_tree(rtree) = k_r ^ add_tree(ltree) = k_l ^
r rtree l ltree t 7! (k; l; r) RL $!_{k+k_l+k_r}$ R g

g

f add_tree(tree) = k^0 ^ t tree RL $!_{k^0}$ R g

g

f add_tree(tree) = k^0 ^ t tree RL $!_{k^0}$ R g

8 Adding a Directed Acyclic Graph with repetitions

$$dag :=$$
$$j8sum; k; (l; r\ : dag)(\ _l;\ \ _r\ : shares)(sum; k;\ \ _l; l;\ \ _r; r)$$

$gy_?$, $g = NULL$

gy_0 , $g = NULL$

$g\ y_{sum}\ (sum; k;\ \ _l; d_l;\ \ _r; d_r)$, $g{:}lock\ \ !_{(sum;k;\ \ _l;l;\ \ _r;r)}\ \ R$

WHERE

R , $(sum; k;\ \ _l; d_l;\ \ _r; d_r){:}9l; r; k; sum_l; sum_r$

```
g:k !   k
g:l !   l
g:r !   r
if g:sum = NULL  then
   sum = sum_l = sum_r = ?
   g:sum = NULL
   l y^l_? d_l   r y^r_? d_r
else
   sum = k + sum_l + sum_r ^
   g:sum !   sum
   l y^l_{sum_l} d_l   r y^r_{sum_r} d_r
```

$$(sum_1; k;\ \ _l; l_1;\ \ _r; r_1)\quad (sum_2; k;\ \ _l; l_2;\ \ _r; r_2)\ ,\ (sum_1\quad sum_2; k;\ \ _l; l_1\quad l_2;\ \ _r; r_1\quad r_2)$$

```
struct  node
f
  lock  l;          //lock
  int   sum;          //Partial sum
  int  k;           //key _value
  struct  node  l;  //left  subtree
  struct  node  r;  //right  subtree
g;
```

n

```
f gyd g
   ?
void  AddDag(struct  node   g, int   ret)f
   if (g = NULL) f
      f gy   ret 7! _ g
         ?
      ret = 0;
      f gy   ret 7! 0 g
         0
      return ;
   g else f
      f gy(? ; k;  l ; ls;  r ; rs)  ret 7! _ g
         ?
      Acquire(g);
      f 9 do; R(? ; k;  l ; dl ;  r ; dr )   do  Hold g; R; do  gy(? ; k;  l ; dl ;  r ; dr )  ret 7! g
                                                                ?
      f 9 vo; dl ; dr ; R(?   vo; k;  l ; ls   dl ;  r ; rs   dr )
      Hold g; R; do  gy(? ; k;  l ; dl ;  r ; dr )  ret 7! _ g
                       ?
      if (g.sum != NULL)  f
         f R(vo; k;  l ; dl ;  r ; dr )  Hold g; R; do  gy(? ; k;  l ; dl ;  r ; dr )  ret 7! _ g
                                                          ?
         ret = g.sum;
         f R(vo; k;  l ; dl ;  r ; dr )  Hold g; R; do  gy(? ; k;  l ; dl ;  r ; dr )  ret 7! g:sum g
                                                          ?
         Release (g)
         f g y(vo; k;  l ; dl ;  r ; dr )  ret 7! g:sum g
             vo
      g else f
         f R(? ; k;  l ; dl ;  r ; dr )  Hold g; R; do  gy(? ; k;  l ; dl ;  r ; dr )  ret 7! _ g
                                                          ?
         f 9 l; r; k; sum l ; sumr ; g:k 7! k   g:l 7! l   g:r 7! r   g:sum = NULL
            sum = suml = sumr = ?
              l       r
            l y dl   r y dr
              ?       ?
            Hold g; R; do  gy(? ; k;  l ; dl ;  r ; dr )  ret 7! _ g
                             ?
         int   lret, rret;
         thr = Spawn (AddDag, (g.r, rret));
         f thr #!  Rr   g:k 7! k   g:l 7! l   g:r 7! r   g:sum = NULL
                                  l
            sum = suml = ?   l y dl
                               ?
            Hold g; R; do  gy(? ; k;  l ; dl ;  r ; dr )  ret 7! _ g
                             ?
         AddDag (g.l, lret);
         f thr #!  Rr   g:k 7! k   g:l 7! l   g:r 7! r   g:sum = NULL
              l
            l y  dl   lret 7! suml
             sum l
```

```
        Hold g; R; d_o   gy_?(? ; k;  _l ; d_l ;  _r ; d_r )   ret 7! _ g
      Join(the);
      f g:k 7! k   g:l 7! l   g:r 7! r   g:sum = NULL
        l y^l_{sum l} d_l   lret 7! sum_l
        l y^r_{sum r} d_r   rret 7! sum_r
        Hold g; R; d_o   gy_?(? ; k;  _l ; d_l ;  _r ; d_r )   ret 7! _ g
      ret = (g.sum = k + sum _l + sum _r);
      f g:k 7! k   g:l 7! l   g:r 7! r
        g:sum 7! k + sum_l + sum_r
        l y^l_{sum l} d_l   lret 7! sum_l
        l y^r_{sum r} d_r   rret 7! sum_r
        Hold g; R; d_o   gy_?(? ; k;  _l ; d_l ;  _r ; d_r )   ret 7! k + sum_l + sum_r g
      f R(k + sum_l + sum_r ; k;  _l ; d_l ;  _r ; d_r )
        Hold g; R; d_o   gy_?(? ; k;  _l ; d_l ;  _r ; d_r )   ret 7! sum^0 g    nn sum 0= k + sum l + sum r
      Release (g);
      f g  y_{sum 0} (sum^0; k;  _l ; d_l ;  _r ; d_r )   ret 7! sum^0 g
    g
  g
g
f 9 sum^0; d; g  y_{sum 0} d   ret 7! sum^0 g
```