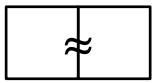
Schorr-Waite


cell



pair


unmarked cell


marked cell

atom


right pointer into Memory


left pointer into Memory


pointer into Memory


null pointer


any pointer

pointers = $\mathbb{N}$

types = { a(tom), p(ointer) }
marks = { m(arked), u(nmarked) }
cells = pointers × types × marks

$*$: pointers $\longrightarrow$ cells : p $\longmapsto$ $[\pi, \tau, \mu]$

$\uparrow$: pointers $\longrightarrow$ pointers : p $\longmapsto$ p$\uparrow$ $\equiv *p_{\pi}$

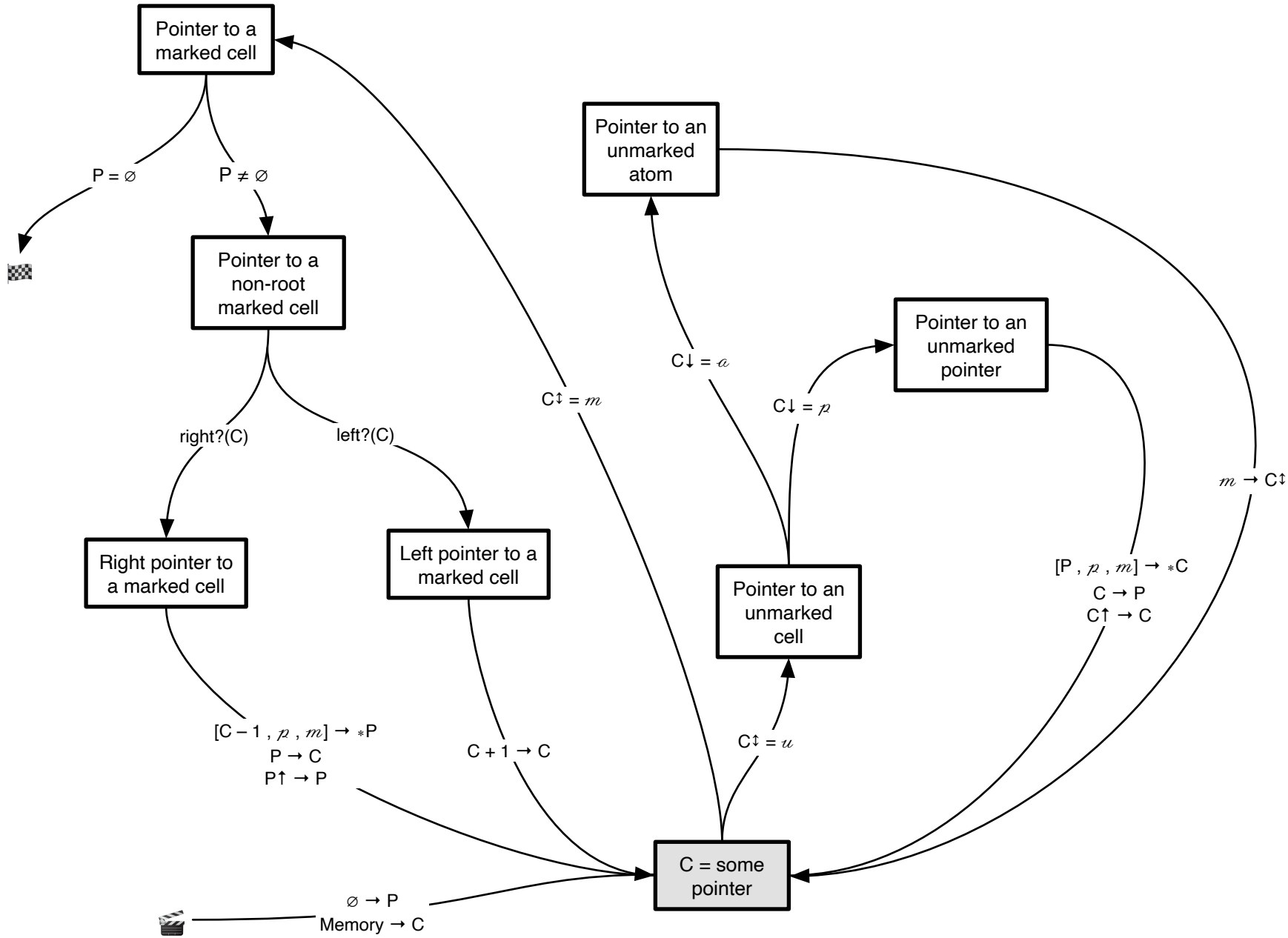
$\downarrow$: pointers $\longrightarrow$ types : p $\longmapsto$ p$\downarrow \equiv *p_{\tau}$

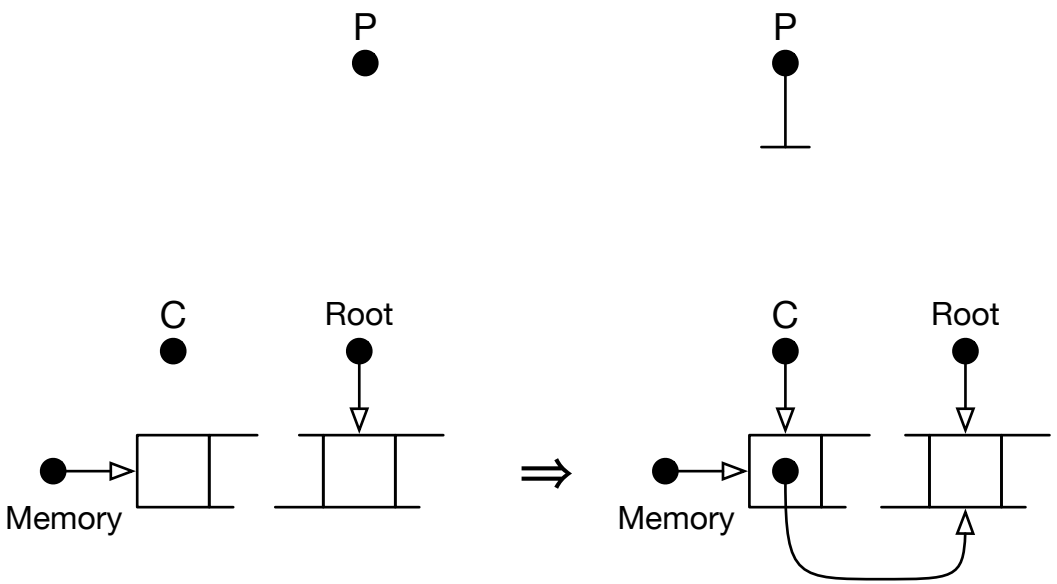
$\updownarrow$: pointers $\longrightarrow$ marks : p $\longmapsto$ p$\updownarrow \equiv *p_{\mu}$

Memory : memory pointer
Root : root pointer

$\varnothing$: null pointer

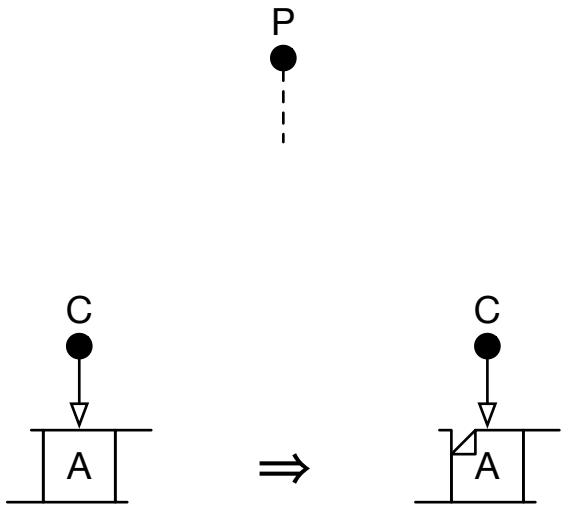
C : current pointer
P : previous pointer


Pointer to a marked cell
P = ∅
P ≠ ∅
Pointer to a non-root marked cell
right?(C)
left?(C)
Right pointer to a marked cell
Left pointer to a marked cell
[C – 1 , p , m] → *P
P → C
P↑ → P
C + 1 → C
C↕ = m
C↓ = a
C↓ = p
Pointer to an unmarked atom
Pointer to an unmarked pointer
Pointer to an unmarked cell
C↕ = u
[P , p , m] → *C
C → P
C↑ → C
m → C↕
C = some pointer
∅ → P
Memory → C



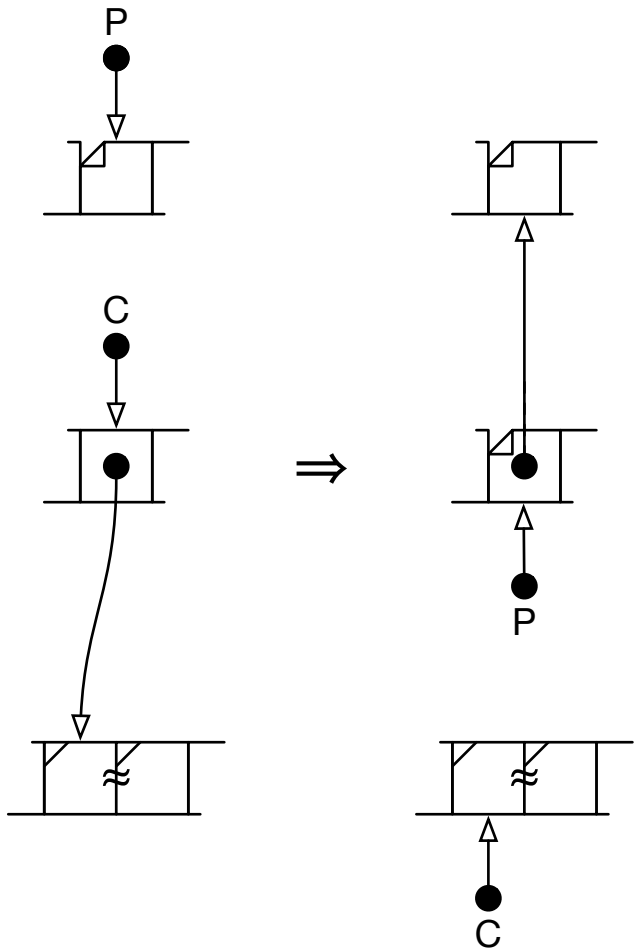
{ [⌀ , Memory } → { P , C }

$(C\updownarrow = u) \wedge (C\downarrow = a)$



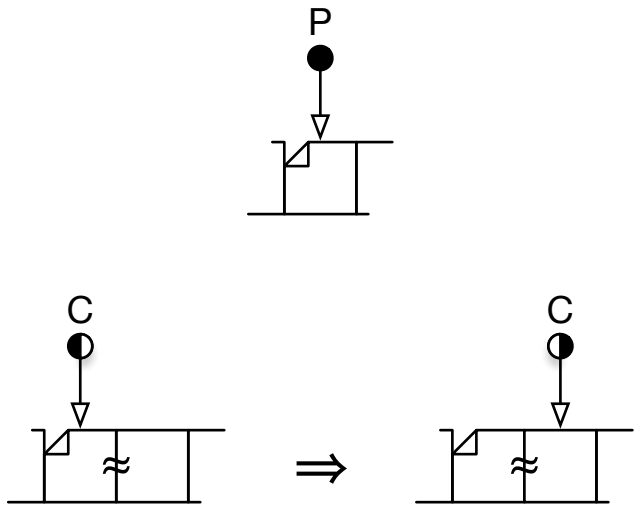
$\{ m \} \rightarrow \{ C\updownarrow \}$

$(C\updownarrow = u) \wedge (C\downarrow = p)$



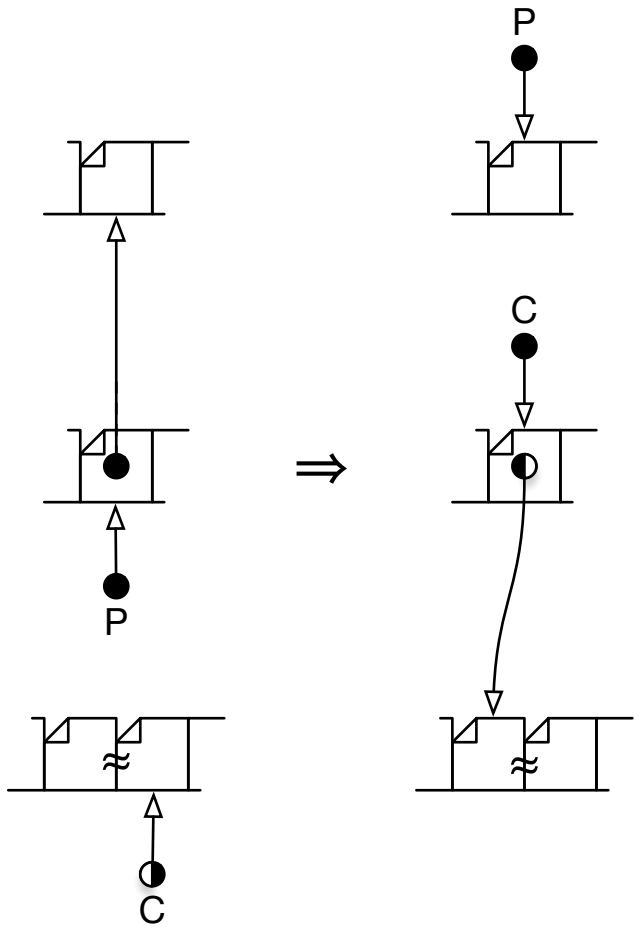
$\{ [P , p , m] , C , C\uparrow \} \rightarrow \{ *C , P , C \}$

$(C\updownarrow = m) \wedge (P \neq \varnothing) \wedge \text{left?}(C)$



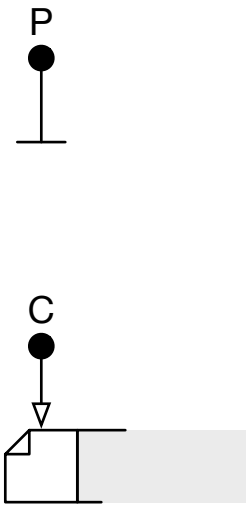
$\{ C + 1 \} \rightarrow \{ C \}$

$(C\updownarrow = m) \wedge (P \neq \varnothing) \wedge \text{right?}(C)$



$\{\,[C-1\,,\,p\,,\,m]\,,\,P\,,\,P\uparrow\,\} \rightarrow \{\,*P\,,\,C\,,\,P\,\}$

$(C\updownarrow = m) \wedge (P = \varnothing)$



```
typedef struct CEL * ptr;
typedef enum {a, p} typ;
typedef enum {m, u} mrk;
typedef struct CEL { ptr P; typ T; mrk M; } cel;

unsigned is_left(ptr);

ptr Memory, Null;

void Schorr_Waite(cel Root)
  { ptr C, C_, P, P_;
    *Memory = Root;                          // *Memory <- Root
    P = Null;                                // P <- null
    for (C = Memory;; )                      // C <- Memory
      if (C->M == u)                         // Cw = u
        if (C->T == a)                       // Cv = a
          C->M = m;                          // Cw = m
        else                                 // Cv = p
          { C_ = C->P;                       // C^
            *C = (cel){ P, p, m };           // *C = [P, p, m]
            P = C;                           // P <- C
            C = C_; }                        // C <- C^
      else                                   // Cw = m
        if (P != Null)                       // P ≠ null
          if (is_left(C))                    // left?(C)
            C += 1;                          // C <- C + 1
          else                               // right?(C)
            { P_ = P->P;                     // P^
              *P = (cel){ C - 1, p, m };     // *P = [C - 1, p, m]
              C = P;                         // C <- P
              P = P_; }                      // P <- P^
        else                                 // P = null
          break; }                           // stop
```