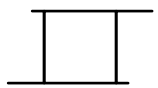
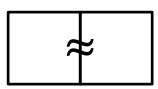
# Jonkers-Schorr-Waite link



cell



pair

chunk

unmarked cell

marked cell

any cell

header

pair pointer into cell storage

chunks, pairs $\subset \mathbb{N}$
chunks $\cap$ pairs = $\varnothing$
pointers = chunks $\cup$ pairs
types = { $\mathscr{a}$(tom), $\mathscr{h}$(eader), $\mathscr{p}$(ointer) }
marks = { $\mathscr{m}$(arked), $\mathscr{u}$(nmarked) }
cells = pointers × types × markers
$*$: pointers $\longleftrightarrow$ cells : p $\longleftrightarrow$ [$\pi$, $\tau$, $\mu$]

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

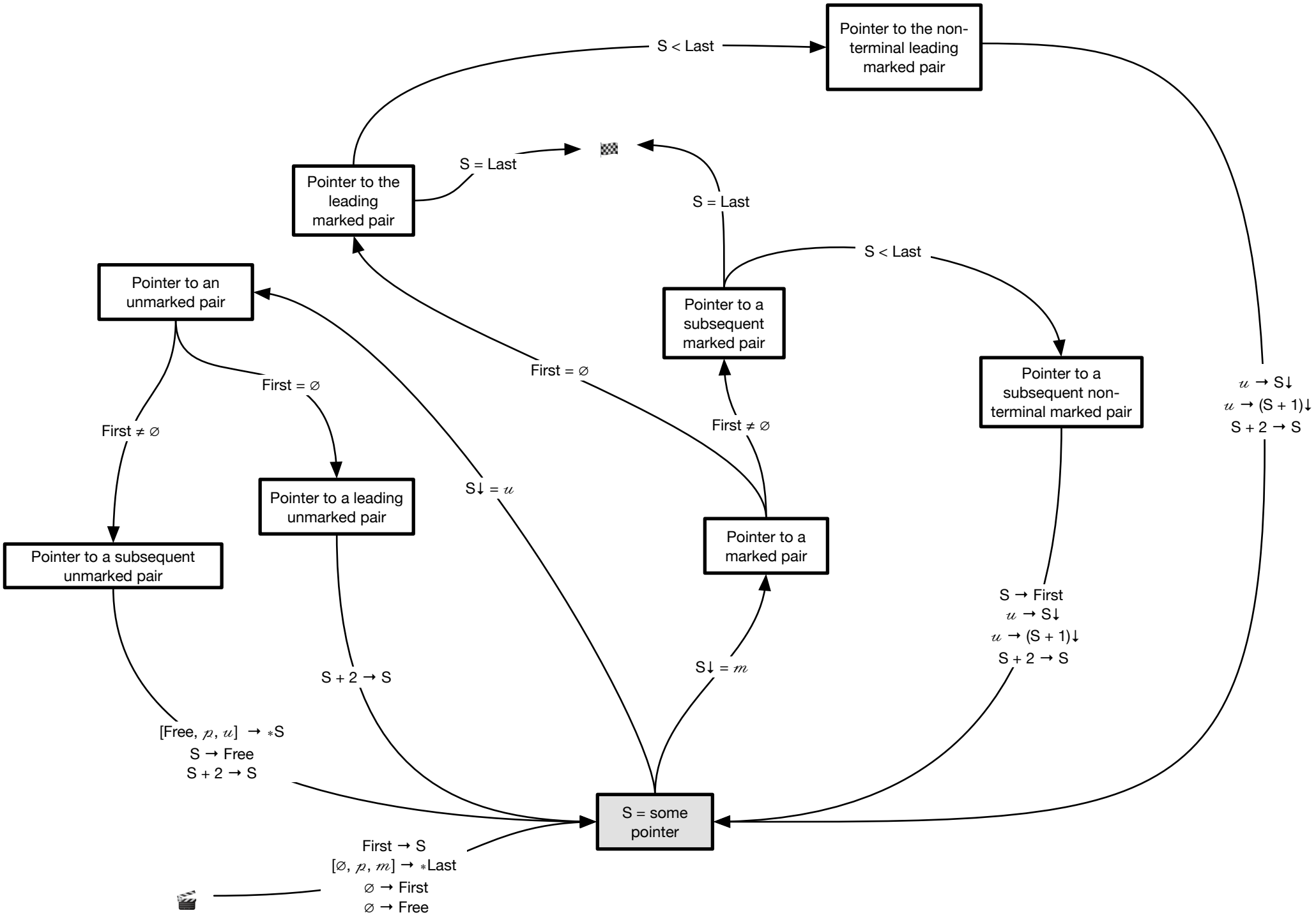
$\updownarrow$: pointers $\longrightarrow$ pointers : p $\longmapsto$ p$\updownarrow$ $\equiv$ $*\mathrm{p}_{\tau}$

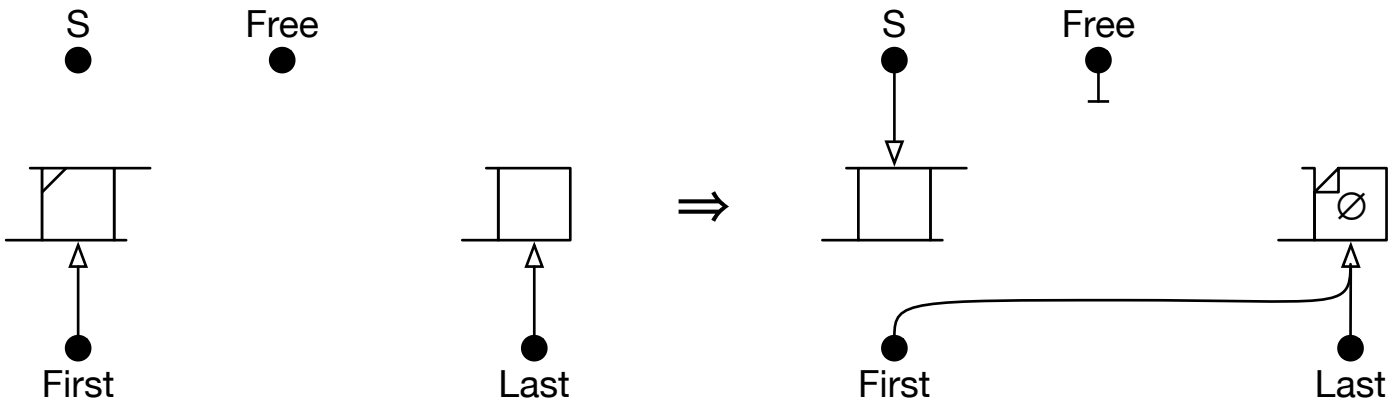
$\downarrow$: pointers $\longrightarrow$ markers : p $\longmapsto$ p$\downarrow$ $\equiv$ $*\mathrm{p}_{\mu}$

S : current pointer

Memory : memory pointer
Free : free pointer

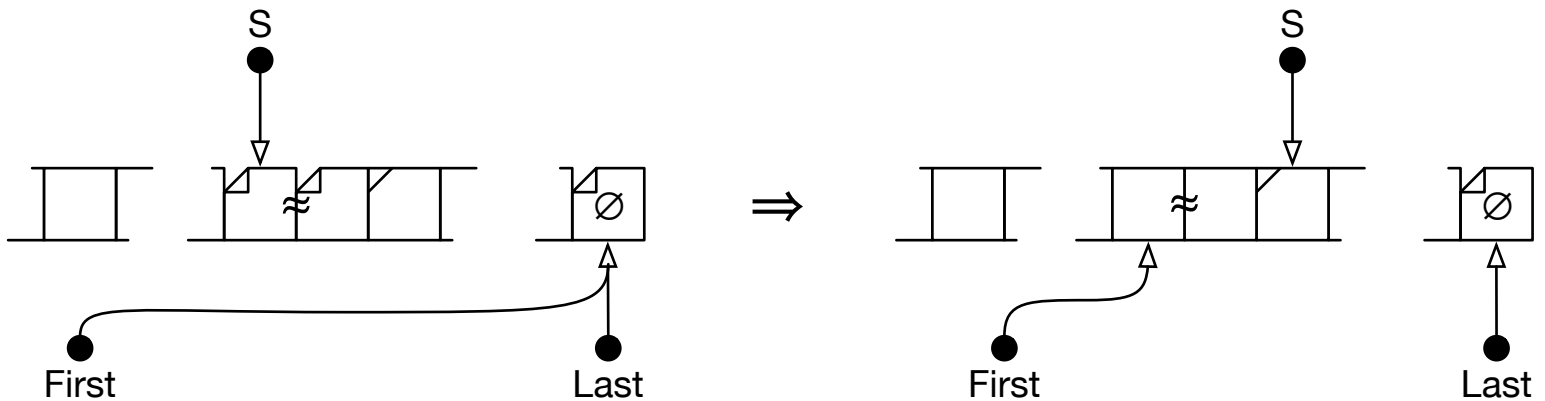
First: first pair pointer
Last: last pair pointer


Pointer to the non-terminal leading marked pair
S < Last
Pointer to the leading marked pair
S = Last
S = Last
Pointer to an unmarked pair
Pointer to a subsequent marked pair
S < Last
First = ∅
Pointer to a subsequent non-terminal marked pair
u → S↓
u → (S + 1)↓
S + 2 → S
First = ∅
First ≠ ∅
First ≠ ∅
Pointer to a leading unmarked pair
S↓ = u
Pointer to a subsequent unmarked pair
Pointer to a marked pair
S → First
u → S↓
u → (S + 1)↓
S + 2 → S
S↓ = m
S + 2 → S
[Free, p, u] → *S
S → Free
S + 2 → S
S = some pointer
First → S
[∅, p, m] → *Last
∅ → First
∅ → Free



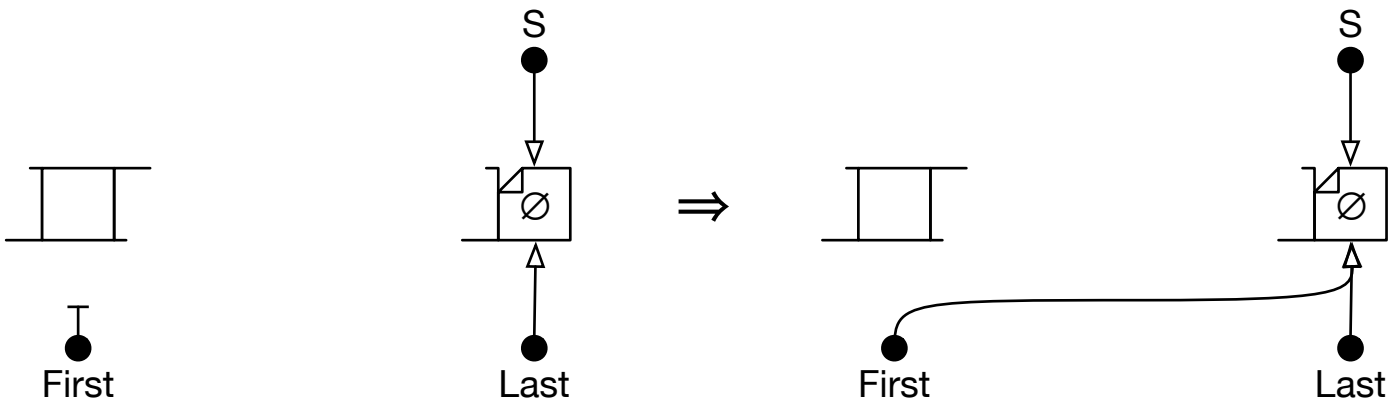

{ First , [∅, $p$, $m$] , Last , ∅ } → { S , ∗Last , First, Free }

$(S\downarrow = m) \wedge (\text{First} = \text{Last}) \wedge (S < \text{Last})$

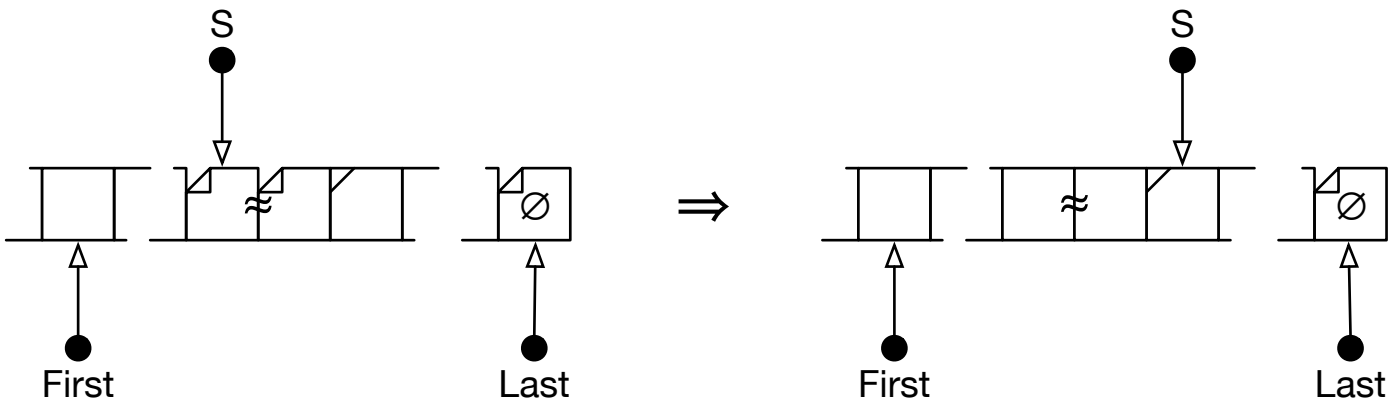



$\approx$

$\{ S , u , u , S + 2 \} \rightarrow \{ \text{First} , S\downarrow , (S + 1)\downarrow , S \}$

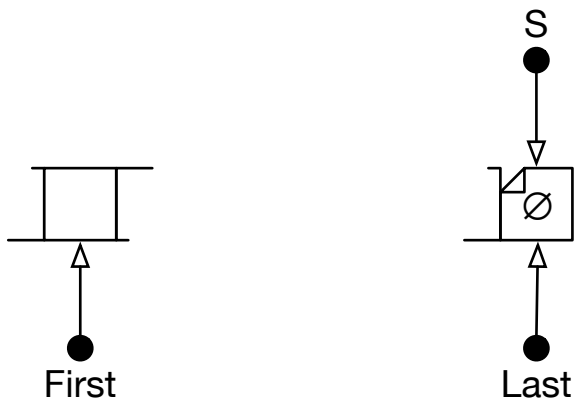

S
First
Last
⇒
S
First
Last

$(S\downarrow = m) \wedge (\text{First} \neq \text{Last}) \wedge (S < \text{Last})$

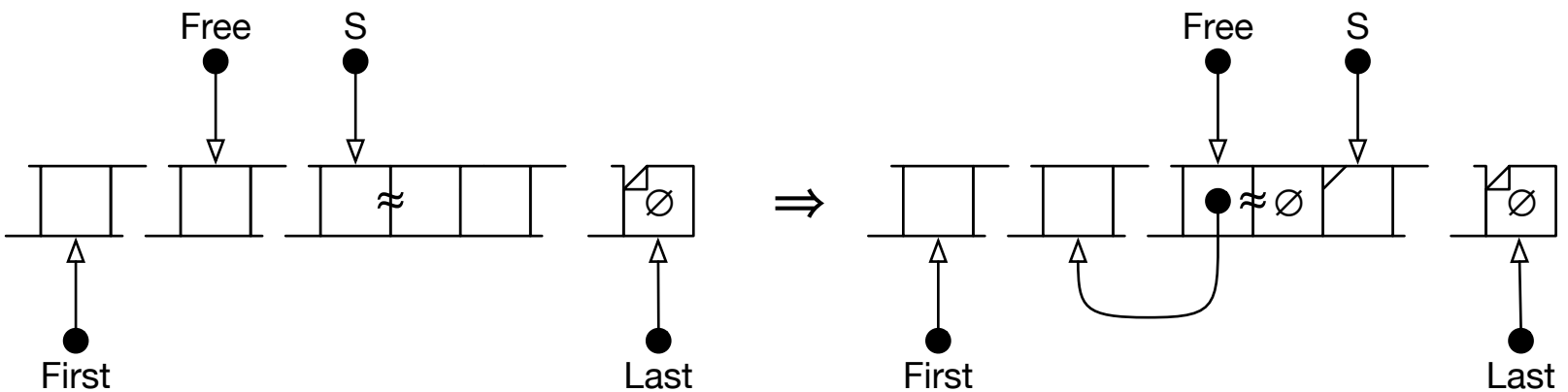



$\{ u , u , S + 2 \} \rightarrow \{ S\downarrow , (S + 1)\downarrow , S \}$

$(S\downarrow = m) \wedge (\text{First} \neq \text{Last}) \wedge (S = \text{Last})$

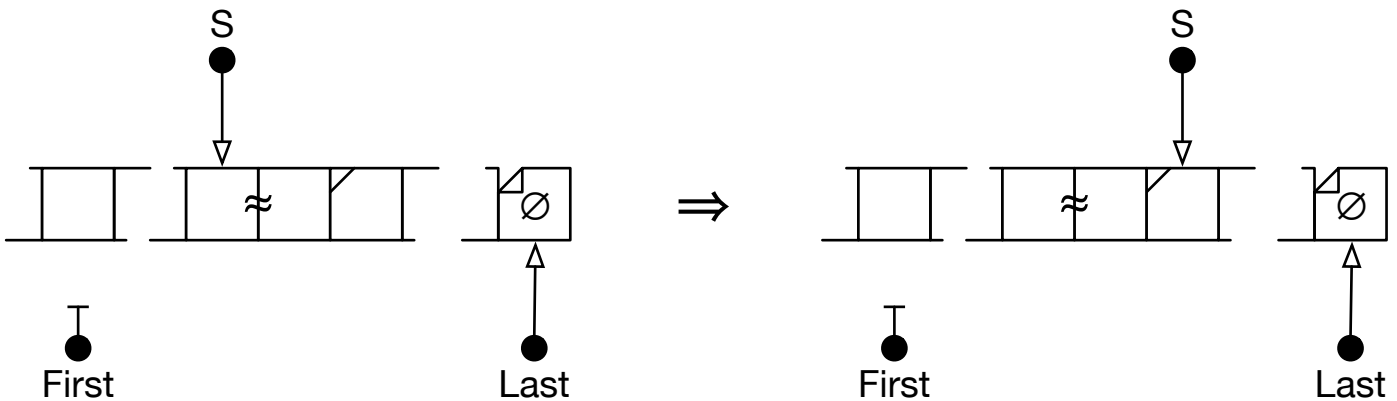


$(S\downarrow = u) \wedge (\text{First} \neq \text{Last})$




$\{ \text{Free} , [\varnothing, p, u] , S , S + 2 \} \rightarrow \{ S\uparrow , *(S + 1) , \text{Free} , S \}$

(S↓ = $\mathcal{U}$) ∧ (First = Last)




{ S + 2 } → { S }

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

ptr First, Free, Last, Null;

void Jonkers_Schorr_Waite_link(void)
  { ptr S;
    S = First;                                  // S <- First
    *Last = (cel){ Null, p, m };                // *Last = [Null, p, m]
    First = Last;                               // First <- Last
    for (Free = Null;;                          // Free <- Null
         S += 2)                                // S <- S + 2
      if (S->M == m)                            // Sv = m
        { if (First == Last)                    // First = Last
            if (S < Last)                       // S < Last
              { First = S;                      // First <- S
                S->M = u;                       // Sv <- u
                (S + 1)->M = u; }               // (S + 1)v <- u
            else                                // S = Last
              break;                            // stop
          else                                  // First ≠ Last
            if (S < Last)                       // S < Last
              { S->M = u;                       // Sv <- u
                (S + 1)->M = u; }               // (S + 1)v <- u
            else                                // S = Last
              break; }                          // stop
      else                                      // Sv = u
        { if (First != Last)                    // First ≠ Last
            { S->P = Free;                      // S^ = Free
              *(S + 1) = (cel){ Null, p, u };   // (S + 1)^ = [Null, p, u]
              Free = S; }                       // Free <- S
          else; }}                              // First = Last
```