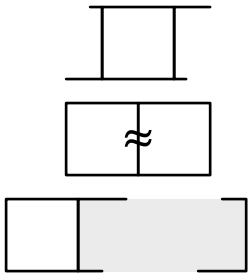
Jonkers-Schorr-Waite unthread



cell

pair

chunk

unmarked cell

marked cell

any cell

$\mathscr{h}$ regular header

$\mathscr{h}^*$ raw header

A atom

pointer into chunk storage
pointer into pair storage
pointer into chunk or pair storage

chunks, pairs $\subset \mathbb{N}$
chunks $\cap$ pairs = $\varnothing$
pointers = chunks $\cup$ pairs

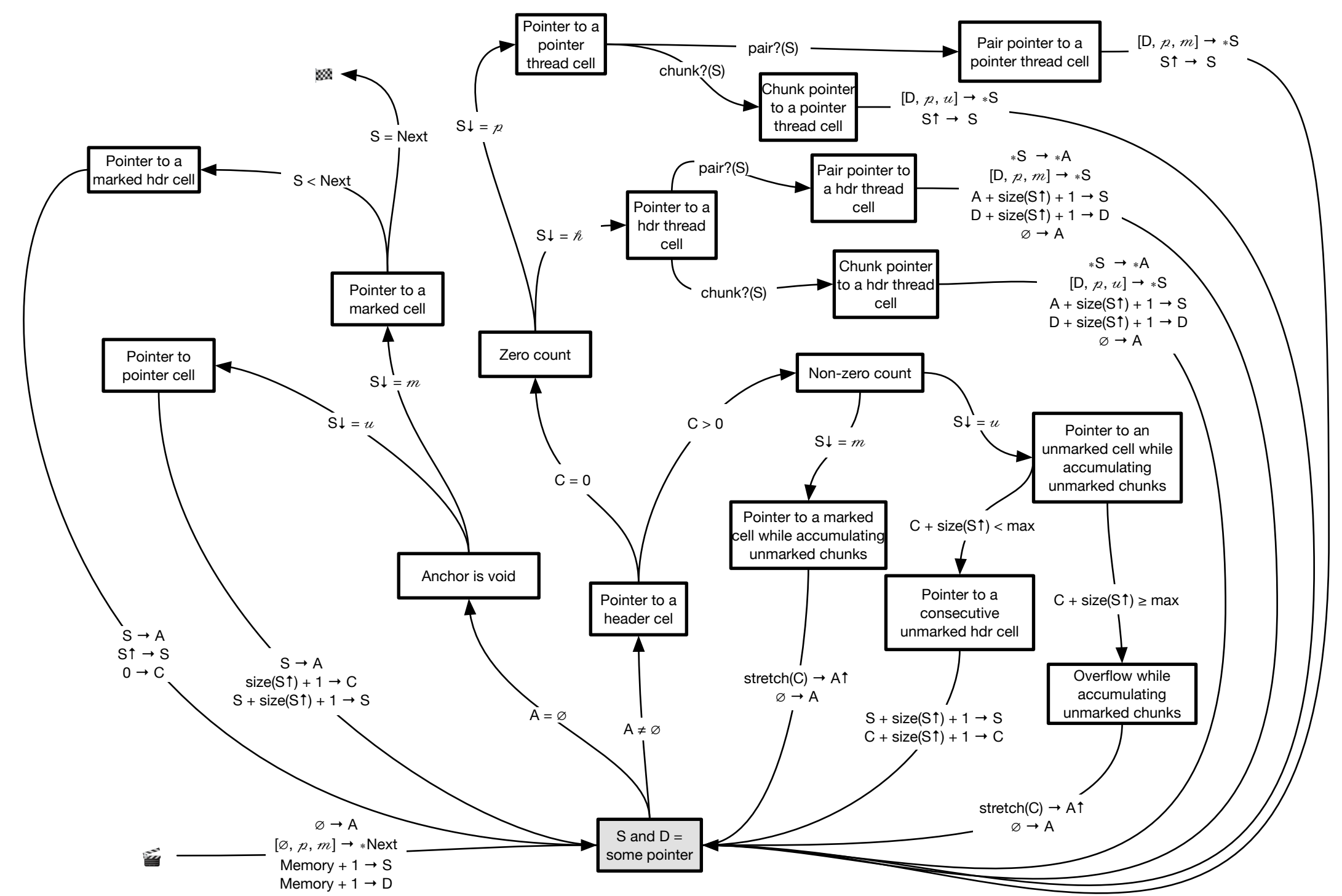
types = { $\mathscr{a}$(tom), $\mathscr{h}$(eader), $\mathscr{p}$(ointer) }
marks = { $\mathscr{m}$(arked), $\mathscr{u}$(nmarked) }
cells = pointers × types × markers

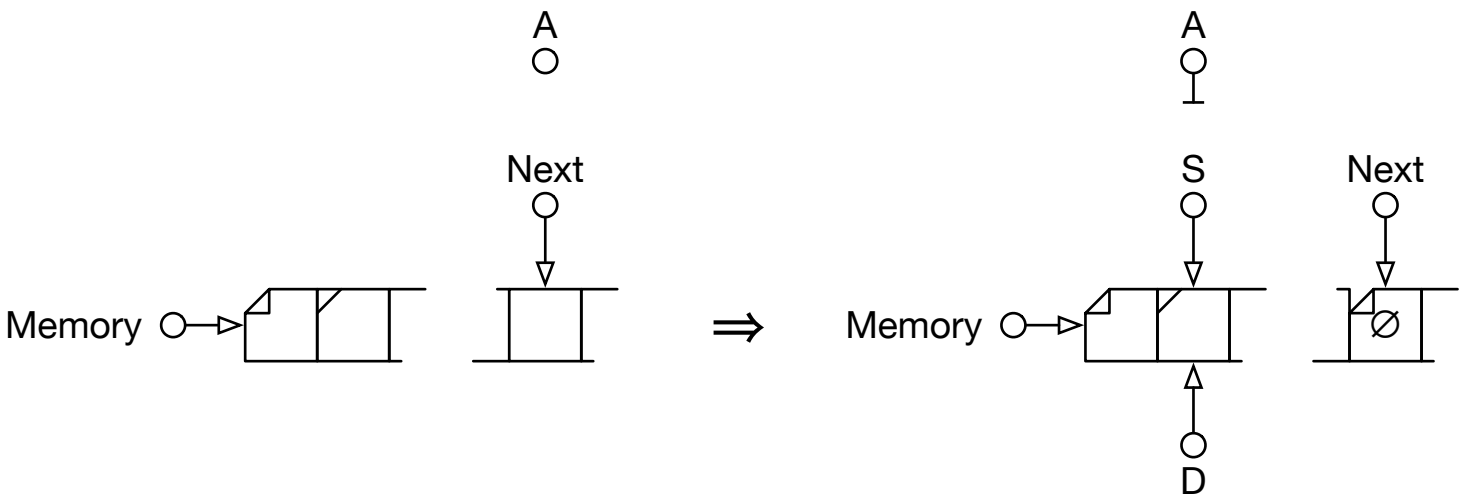
$*$: pointers $\longleftrightarrow$ cells : $p \longleftrightarrow [\pi, \tau, \mu]$
$\uparrow$: pointers $\longrightarrow$ pointers : $p \longmapsto p\uparrow \equiv *p_{\pi}$
$\updownarrow$: pointers $\longrightarrow$ types : $p \longmapsto p\updownarrow \equiv *p_{\tau}$
$\downarrow$: pointers $\longrightarrow$ markers : $p \longmapsto p\downarrow \equiv *p_{\mu}$

chunk? : pointer $\longrightarrow$ boolean
pair? : pointer $\longrightarrow$ boolean
size : pointer $\longrightarrow \mathbb{N}$
stretch : $\mathbb{N} \longrightarrow$ pointers

Memory : memory pointer
Next : next free chunk pointer

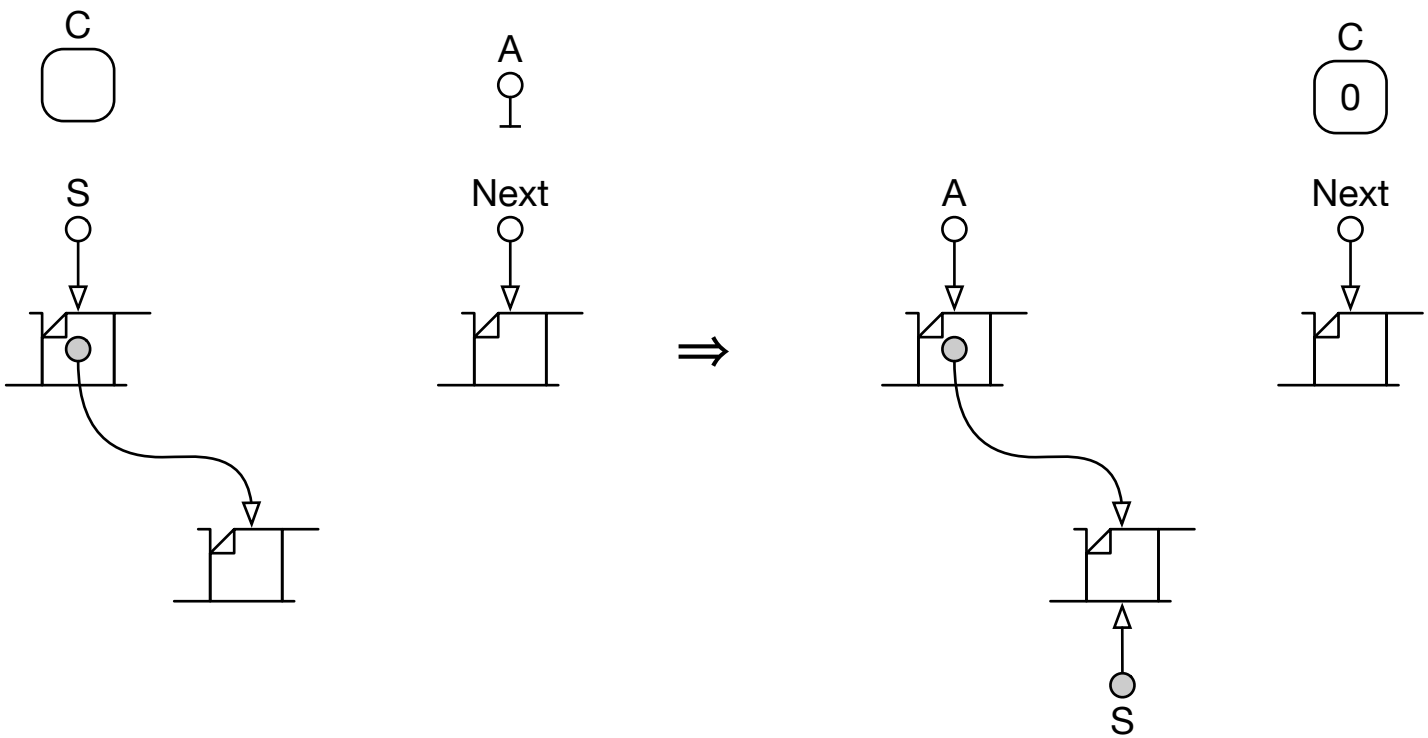
A : anchor pointer
D : destination pointer
S : source pointer
C: counter


Pointer to a pointer thread cell
pair?(S)
Pair pointer to a pointer thread cell
[D, 𝑝, 𝑚] → *S
S↑ → S
chunk?(S)
Chunk pointer to a pointer thread cell
[D, 𝑝, 𝑢] → *S
S↑ → S
S↓ = 𝑝
S = Next
Pointer to a marked hdr cell
S < Next
Pointer to a hdr thread cell
pair?(S)
Pair pointer to a hdr thread cell
*S → *A
[D, 𝑝, 𝑚] → *S
A + size(S↑) + 1 → S
D + size(S↑) + 1 → D
∅ → A
S↓ = ℎ
chunk?(S)
Chunk pointer to a hdr thread cell
*S → *A
[D, 𝑝, 𝑢] → *S
A + size(S↑) + 1 → S
D + size(S↑) + 1 → D
∅ → A
Pointer to a marked cell
Zero count
Pointer to pointer cell
S↓ = 𝑚
S↓ = 𝑢
Non-zero count
C > 0
S↓ = 𝑚
S↓ = 𝑢
Pointer to an unmarked cell while accumulating unmarked chunks
C = 0
Pointer to a marked cell while accumulating unmarked chunks
C + size(S↑) < max
Anchor is void
Pointer to a header cel
Pointer to a consecutive unmarked hdr cell
C + size(S↑) ≥ max
S → A
S↑ → S
0 → C
S → A
size(S↑) + 1 → C
S + size(S↑) + 1 → S
A = ∅
A ≠ ∅
stretch(C) → A↑
∅ → A
S + size(S↑) + 1 → S
C + size(S↑) + 1 → C
Overflow while accumulating unmarked chunks
stretch(C) → A↑
∅ → A
∅ → A
[∅, 𝑝, 𝑚] → *Next
Memory + 1 → S
Memory + 1 → D
S and D = some pointer



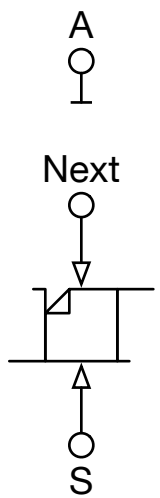
{ ∅ , [∅, $\hbar$, m] , Memory + 1 , Memory + 1 } → { A , ∗Next , S , D }

(A = ∅) ∧ (S↓ = m) ∧ (S < Next)

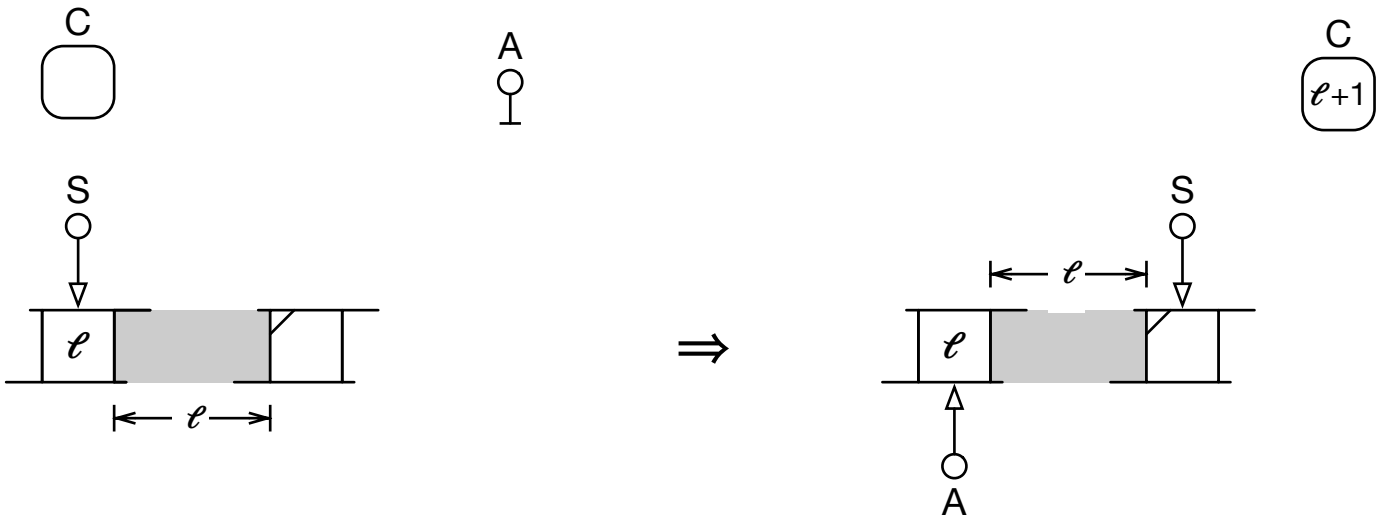


{ S , S↑ , 0 } → { A , S , C }

$(A = \varnothing) \wedge (S\downarrow = m) \wedge (S = \text{Next})$

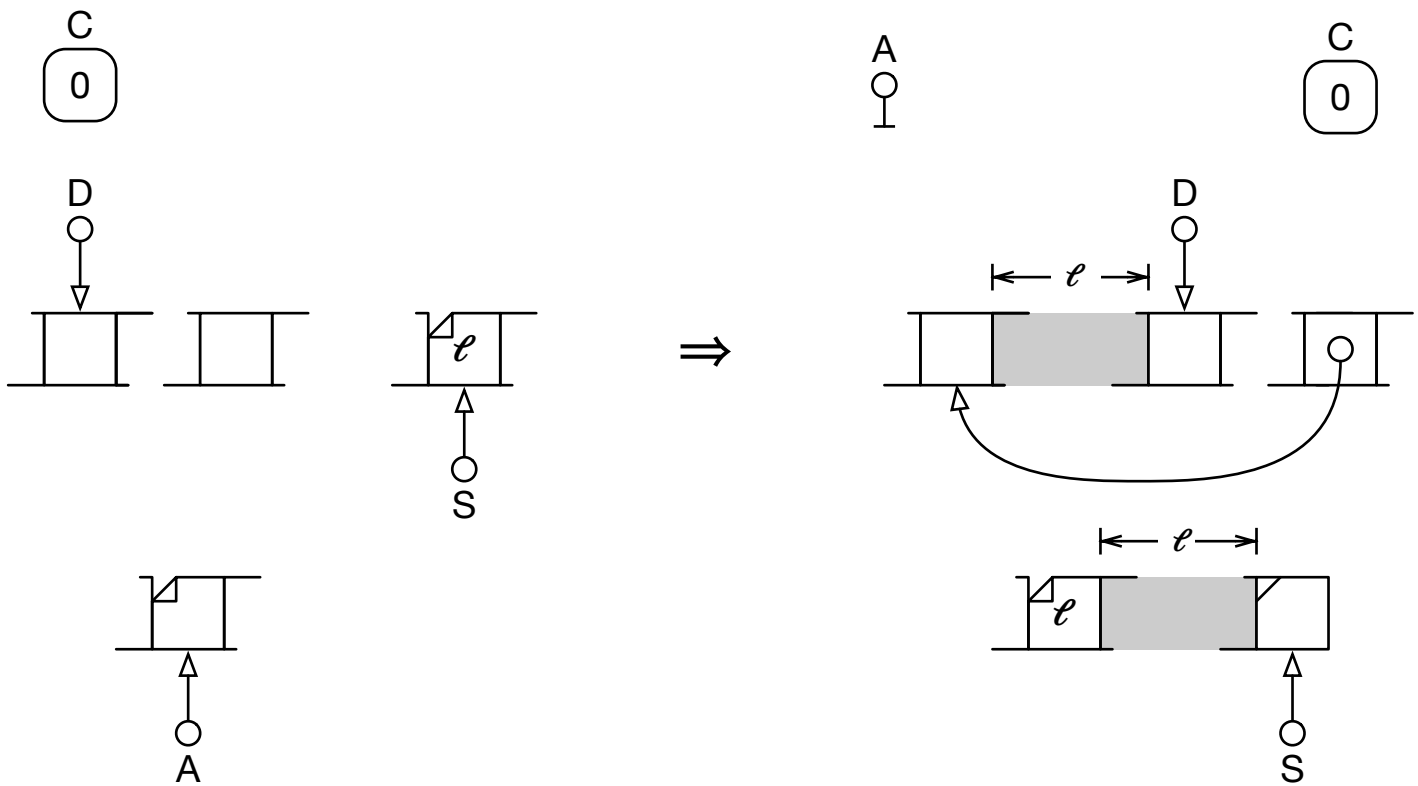


$(A = \varnothing) \wedge (S\downarrow = u)$



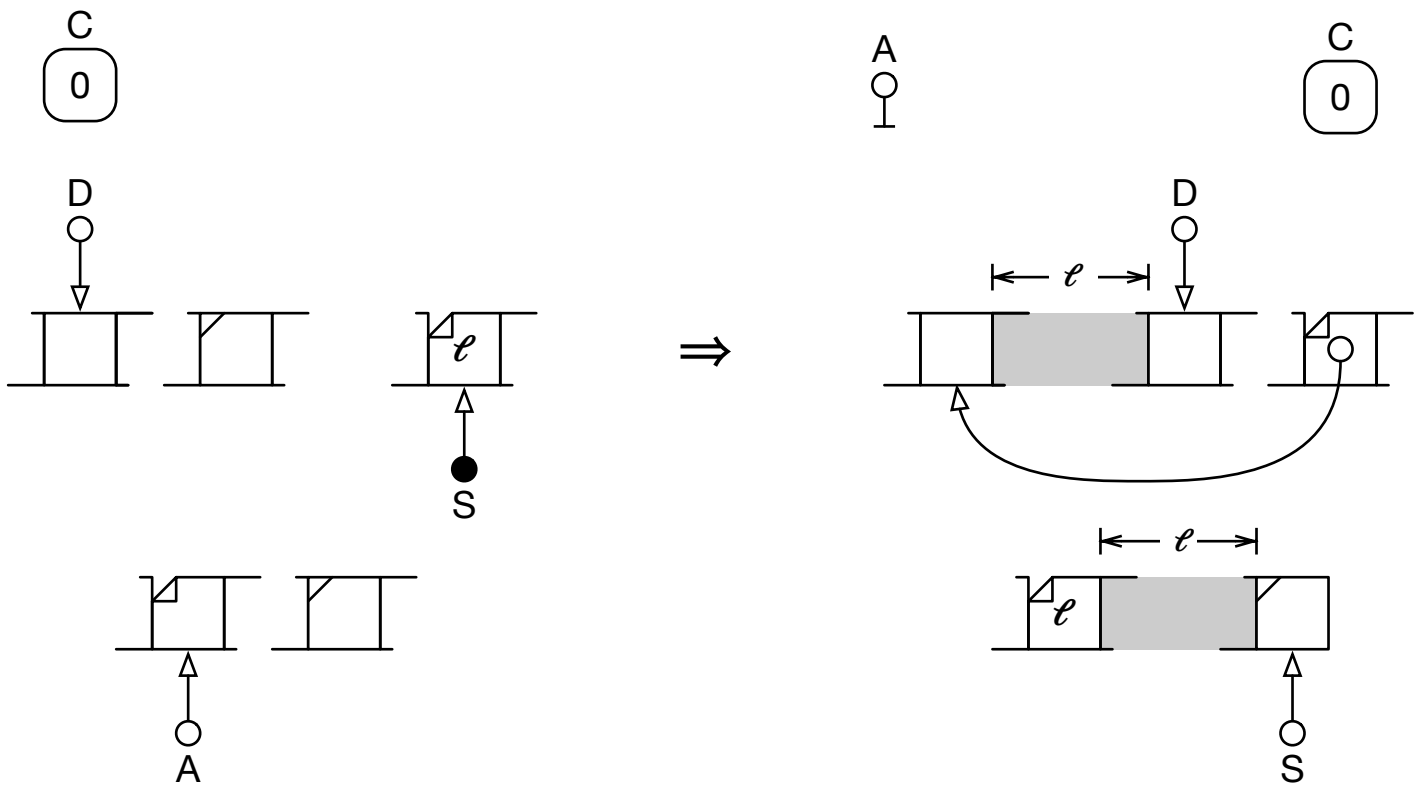
$\{ S , size(S\uparrow) + 1 , S + size(S\uparrow) + 1 \} \rightarrow \{ A , C , S\}$

$(A \neq \varnothing) \wedge (C = 0) \wedge (S\updownarrow = \hbar) \wedge \text{chunk?}(S)$



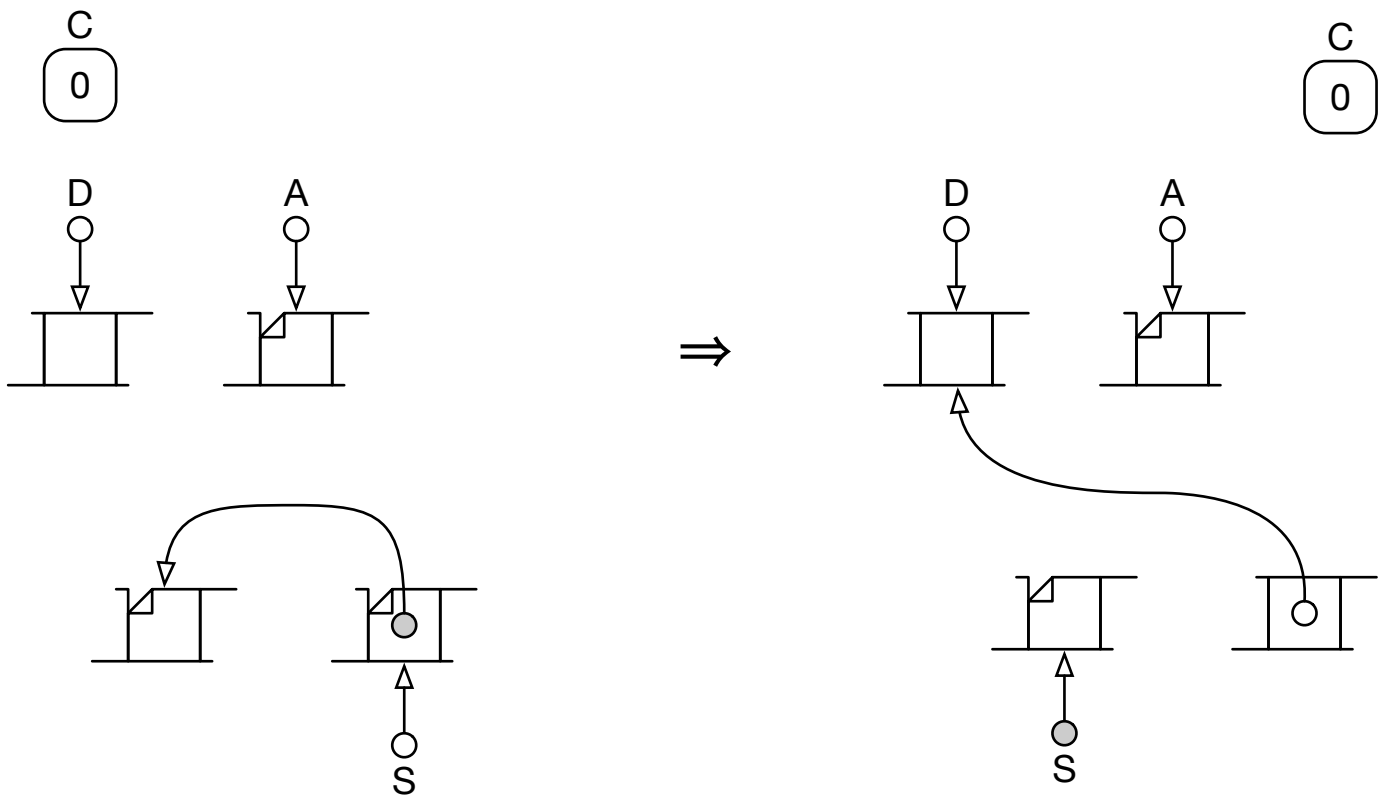
$\{ *S , [D, p, u] , A + \text{size}(S\uparrow) + 1 , D + \text{size}(S\uparrow) + 1 , \varnothing \} \rightarrow \{ *A , *S , S , D , A \}$

$(A \neq \varnothing) \wedge (C = 0) \wedge (S\updownarrow = \hbar) \wedge \text{pair?}(S)$



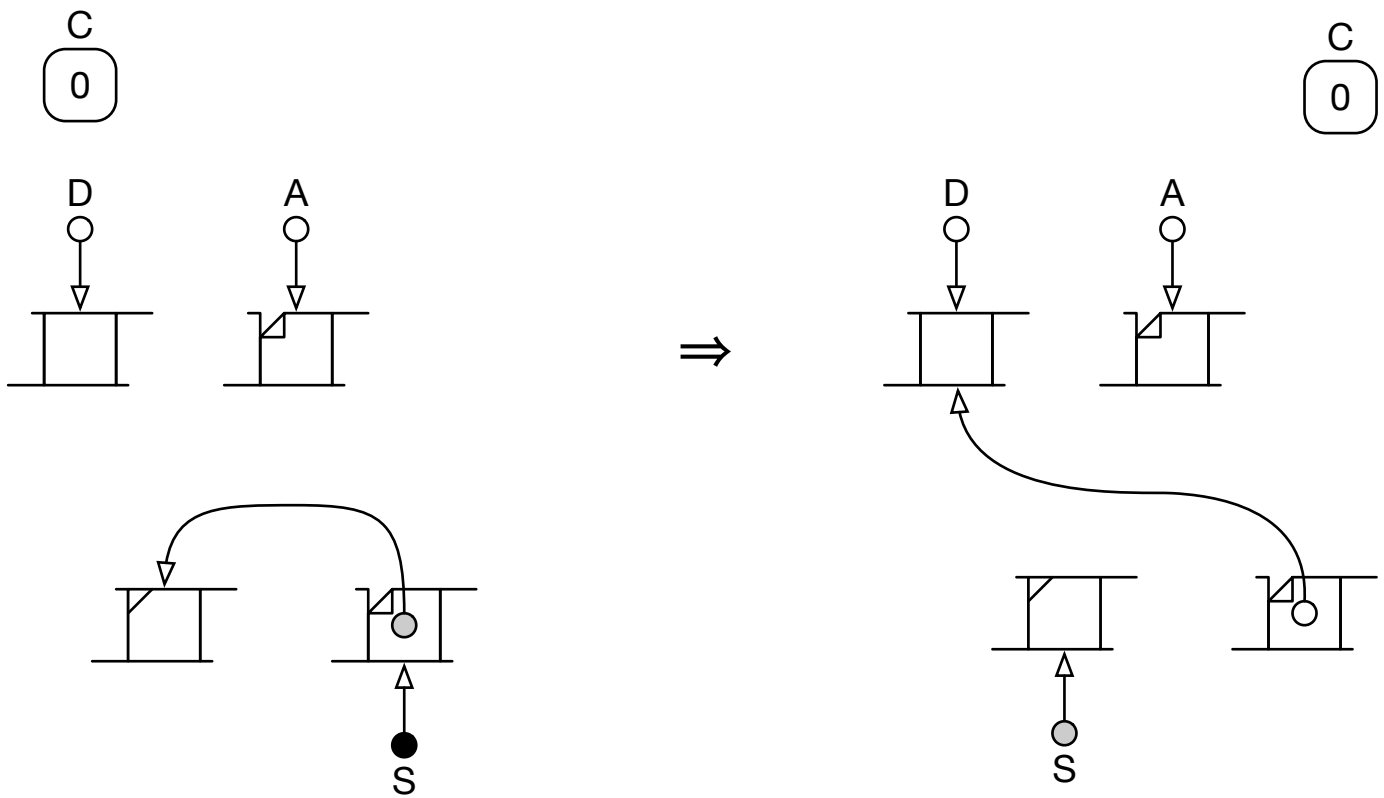
$\{ *S , [D, p, m] , A + \text{size}(S\uparrow) + 1 , D + \text{size}(S\uparrow) + 1 , \varnothing \} \rightarrow \{ *A , *S , S , D , A \}$

$(A \neq \varnothing) \wedge (C = 0) \wedge (S\updownarrow = \mathcal{p}) \wedge \text{chunk?}(S)$



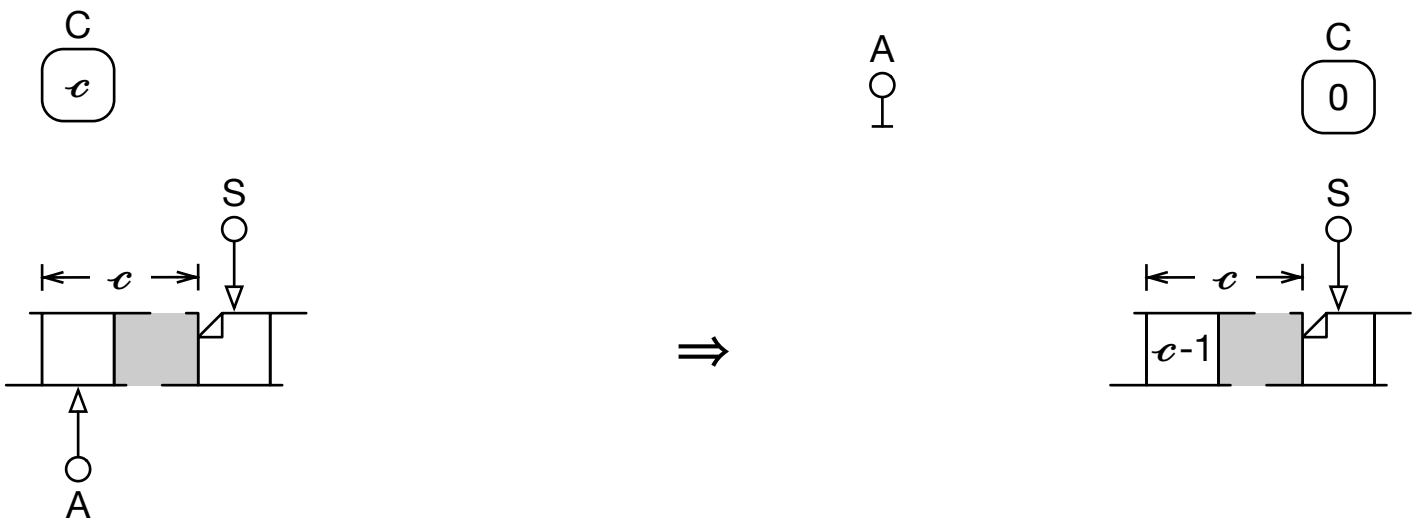
$\{ [D, \mathcal{p}, \mathcal{u}] , S\uparrow \} \rightarrow \{ *S , S \}$

$(A \neq \varnothing) \wedge (C = 0) \wedge (S\updownarrow = p) \wedge \text{pair?}(S)$



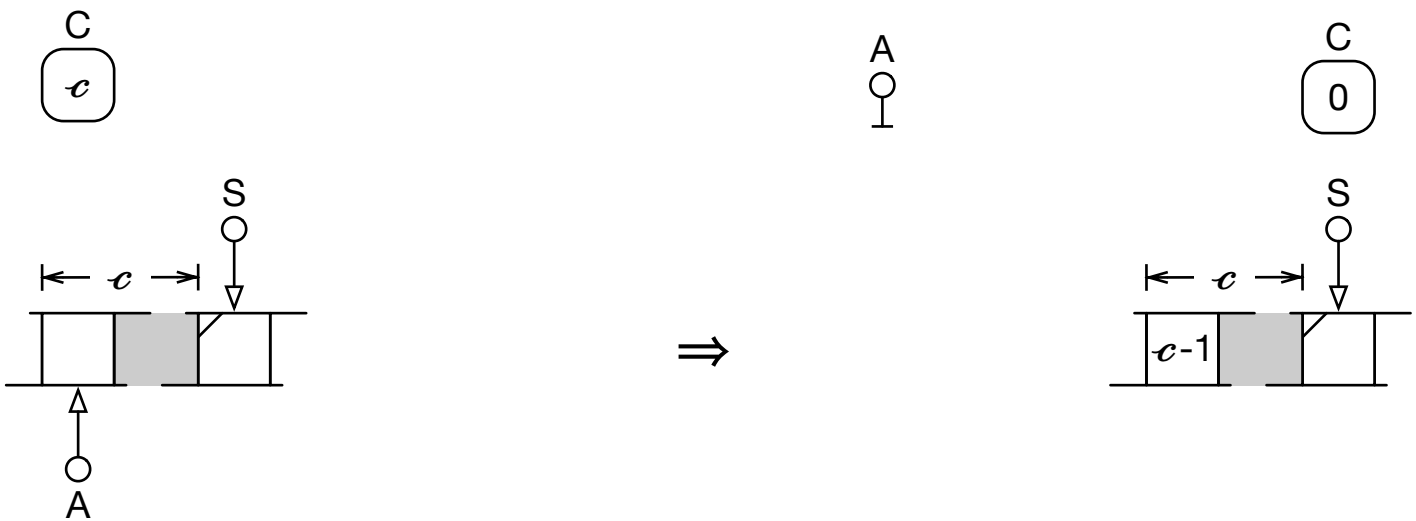
$\{ [D, p, m] , S\uparrow \} \rightarrow \{ *S , S \}$

$(A \neq \varnothing) \wedge (C > 0) \wedge (S\downarrow = m)$



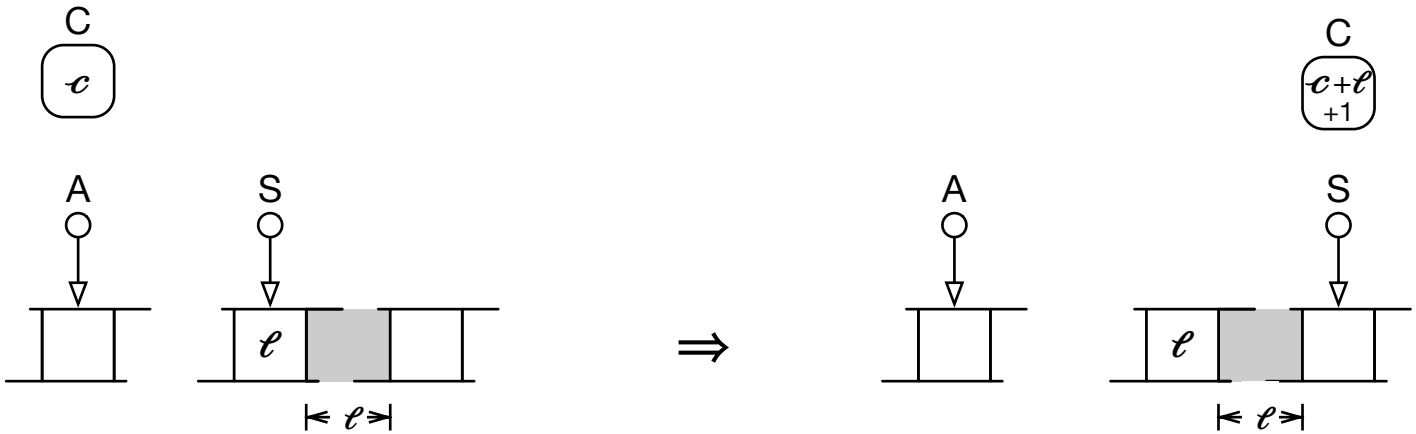
{ stretch(C), ∅ , 0 } → { A↑ , A , C }

$(A \neq \varnothing) \wedge (C > 0) \wedge (S\downarrow = u) \wedge (C + \text{size}(S\uparrow) \geq \text{max})$



$\{ \text{stretch}(C) , \varnothing , 0 \} \rightarrow \{ A\uparrow , A , C \}$

$(A \neq \varnothing) \wedge (C > 0) \wedge (S\downarrow = u) \wedge (C + \text{size}(S\uparrow) < \text{max})$



$\{ S + \text{size}(S\uparrow) + 1 , C + \text{size}(S\uparrow) + 1 \} \rightarrow \{ S , C \}$

```
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;

unsigned Max
ptr Memory, Free, Null;

unsigned is_chunk(ptr), size(ptr);
ptr stretch(unsigned);

void Jonkers_Schorr_Waite_unthread(void)
  { ptr A, D, S, S_;
    unsigned C, L1;
    A = Null;                     // A <- Null
    *Next = (cel){Null, h, m };   // *Next <- [Null, h, m]
    for (S = D = Memory + 1;;)    // S <- D <- Memory + 1
      { S_ = S->P;                // S^
        if (A == Null)            // A = Null
          if (S->M == m)          // Sv = m
            if (S < Next)         // S < Next
              { A = S;            // A <- S
                S = S_;           // S <- S^
                C = 0; }          // C <- 0
            else                  // S = Next
              break;              // stop
          else                    // Sv = u
            { L1 = size(S_) + 1;// size(S^) + 1
              A = S;              // A <- S
              C = L1;             // C <- size(S^) + 1
              S += L1; }          // S <- S + size(S^) + 1
        else                      // A ≠ Null
          if (C == 0)             // C = 0
            if (S->T == h)        // Sw = h
              { L1 = size(S_) + 1;// size(S^) + 1
                *A = *S;          // *A <- *S
                if (is_chunk(S))  // chunk?(S)
                  *S = (cel){ D, p, u };// *S <- [D, p, u]
                else              // pair?(S)
                  *S = (cel){ D, p, m };// *S <- [D, p, m]
                S = A + L1;       // S <- A + size(S^) + 1
                D += L1;          // D <- D + size(S^) + 1
                A = Null; }       // A <- Null
            else                  // Sw = p
              { if (is_chunk(S))  // chunk?(S)
                  *S = (cel){ D, p, u };// *S <- [D, p, u]
                else              // pair?(S)
                  *S = (cel){ D, p, m };// *S <- [D, p, m]
                S = S_; }         // S <- S^
          else                    // C > 0
            if (S->M == m)        // Sv = m
              { A->P = stretch(C);// A^ <- stretch(C)
                A = Null;         // A <- Null
                C = 0; }          // C = 0
            else                  // Sv = u
              { L1 = size(S_) + 1;// size(S^) + 1
                if (C + L1 > Max) // C + size(S^) ≥ Max
                  { A->P = stretch(C);// A^ <- stretch(C)
                    A = Null;     // A <- Null
                    C = 0; }      // C = 0
                else              // C + size(S^) ≤ Max
                  { S += L1;      // S <- S + size(S^) + 1
                    C += L1; }}}} // C <- C + size(S^) + 1
```