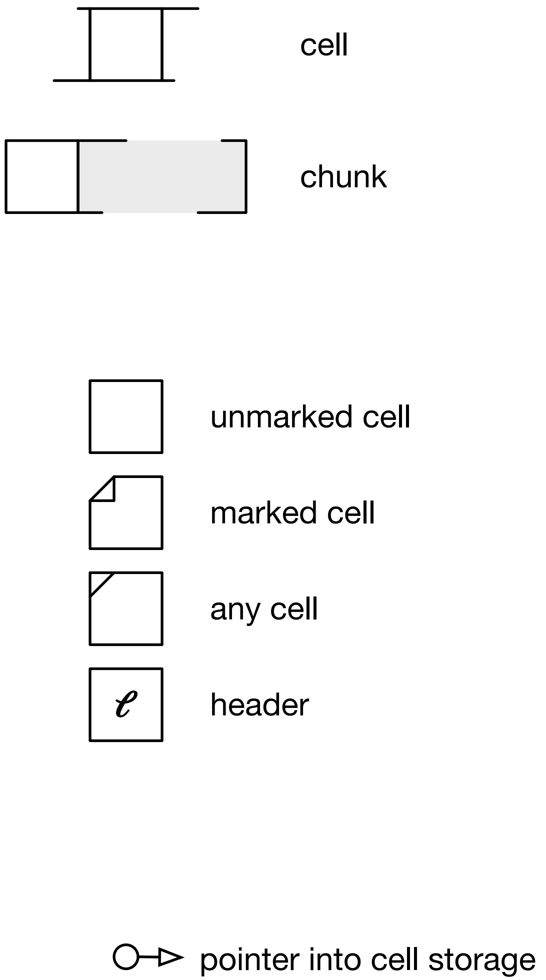
Jonkers unthread



cell

chunk

unmarked cell

marked cell

any cell

header

pointer into cell storage

pointers = $\mathbb{N}$

flags = { $\mathscr{a}$(tom), $\mathscr{m}$(arked), $\mathscr{u}$(nmarked) }
cells = pointers × flags

$*$: pointers $\longleftrightarrow$ cells : $p \longleftrightarrow [\pi , \varphi]$

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

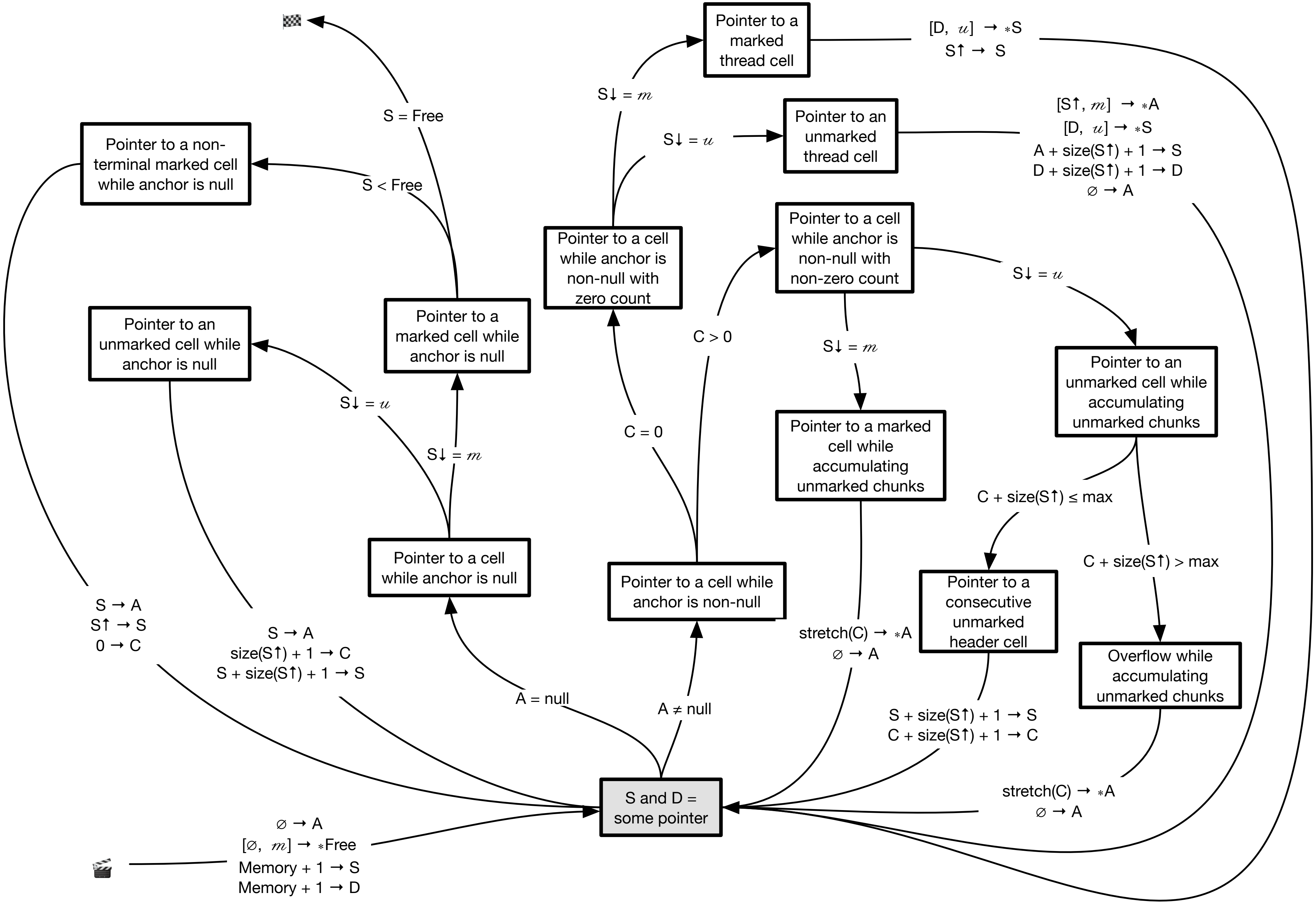
$\downarrow$: pointers $\longrightarrow$ flags : $p \longmapsto p\!\downarrow \equiv *p_{\varphi}$

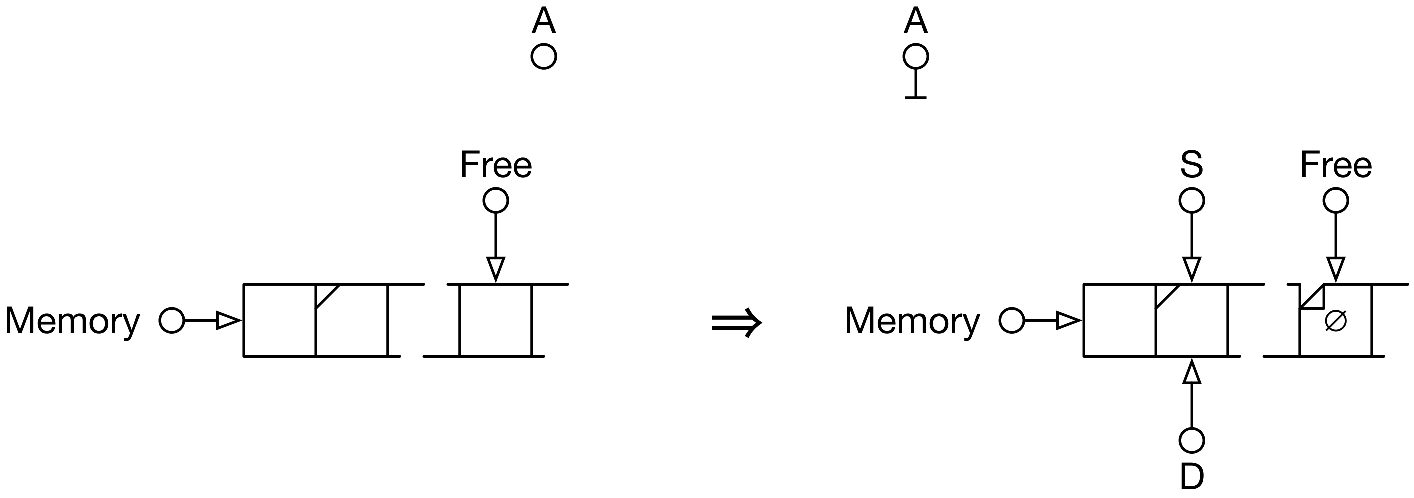
regular? : cells ⟶ boolean
raw? : cells ⟶ boolean
size : cells ⟶ $\mathbb{N}$
stretch : $\mathbb{N}$ ⟶ pointers

Memory : Memory pointer
Free : Free pointer

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

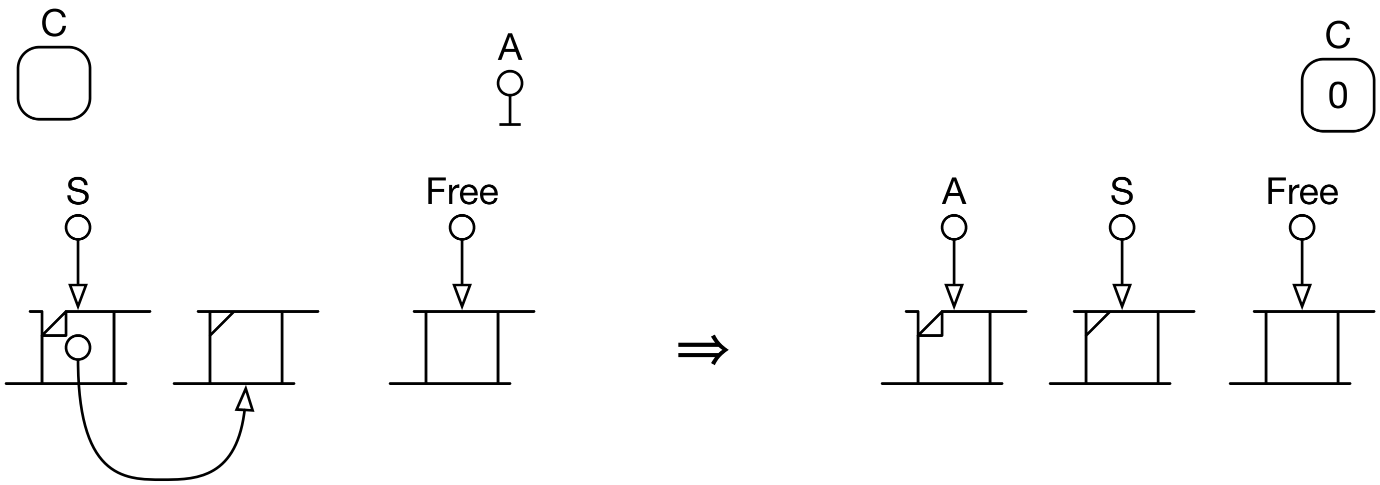
C : counter


Pointer to a marked thread cell
[D, u] → *S
S↑ → S
S↓ = m
Pointer to an unmarked thread cell
S↓ = u
[S↑, m] → *A
[D, u] → *S
A + size(S↑) + 1 → S
D + size(S↑) + 1 → D
∅ → A
S = Free
S < Free
Pointer to a non-terminal marked cell while anchor is null
Pointer to a cell while anchor is non-null with zero count
Pointer to a cell while anchor is non-null with non-zero count
S↓ = u
Pointer to an unmarked cell while anchor is null
Pointer to a marked cell while anchor is null
C > 0
S↓ = m
Pointer to an unmarked cell while accumulating unmarked chunks
S↓ = u
S↓ = m
C = 0
Pointer to a marked cell while accumulating unmarked chunks
C + size(S↑) ≤ max
Pointer to a cell while anchor is null
Pointer to a cell while anchor is non-null
C + size(S↑) > max
S → A
S↑ → S
0 → C
S → A
size(S↑) + 1 → C
S + size(S↑) + 1 → S
stretch(C) → *A
∅ → A
Pointer to a consecutive unmarked header cell
Overflow while accumulating unmarked chunks
A = null
A ≠ null
S + size(S↑) + 1 → S
C + size(S↑) + 1 → C
S and D = some pointer
stretch(C) → *A
∅ → A
∅ → A
[∅, m] → *Free
Memory + 1 → S
Memory + 1 → D



$\{ \varnothing , [\varnothing, \mathcal{m}] , \text{Memory} + 1 , \text{Memory} + 1 \} \to \{ A , {*}\text{Free} , S , D \}$

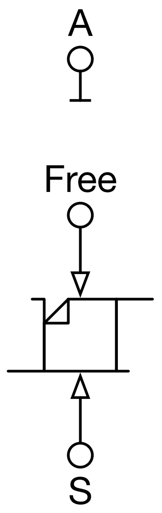
$(A = \varnothing) \wedge (S{\downarrow} = \mathscr{m}) \wedge (S < \text{Free})$



$\{ S , S{\uparrow} , 0 \} \rightarrow \{ A , S , C \}$

Anchor is void

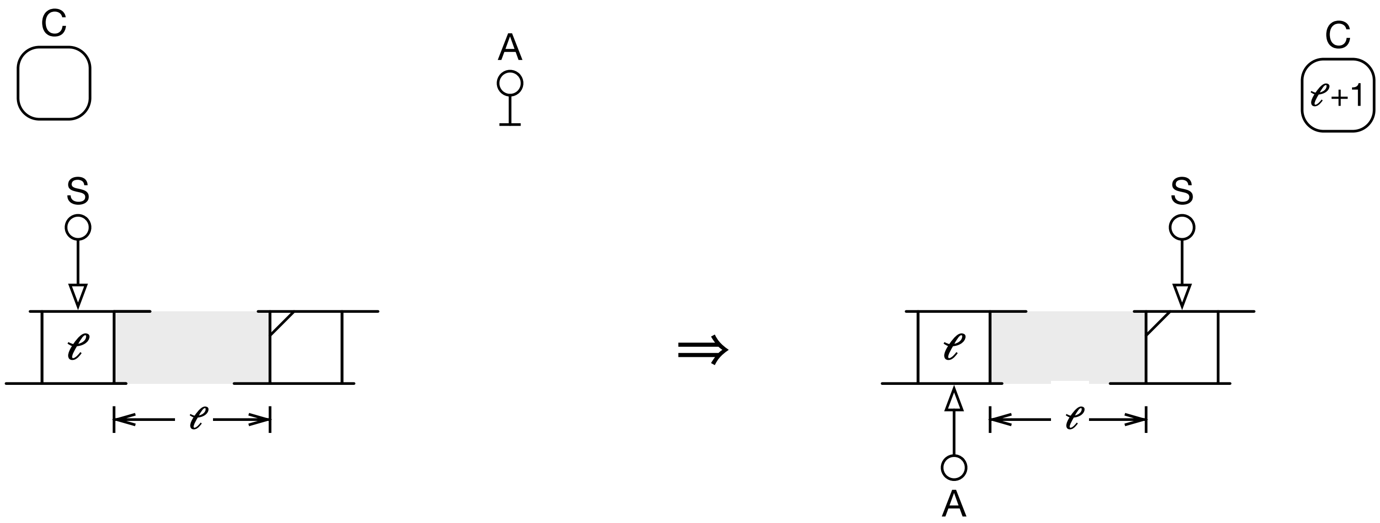
$(A = \varnothing) \wedge (S\downarrow = \mathscr{m}) \wedge (S = \text{Free})$



🏁

Stop

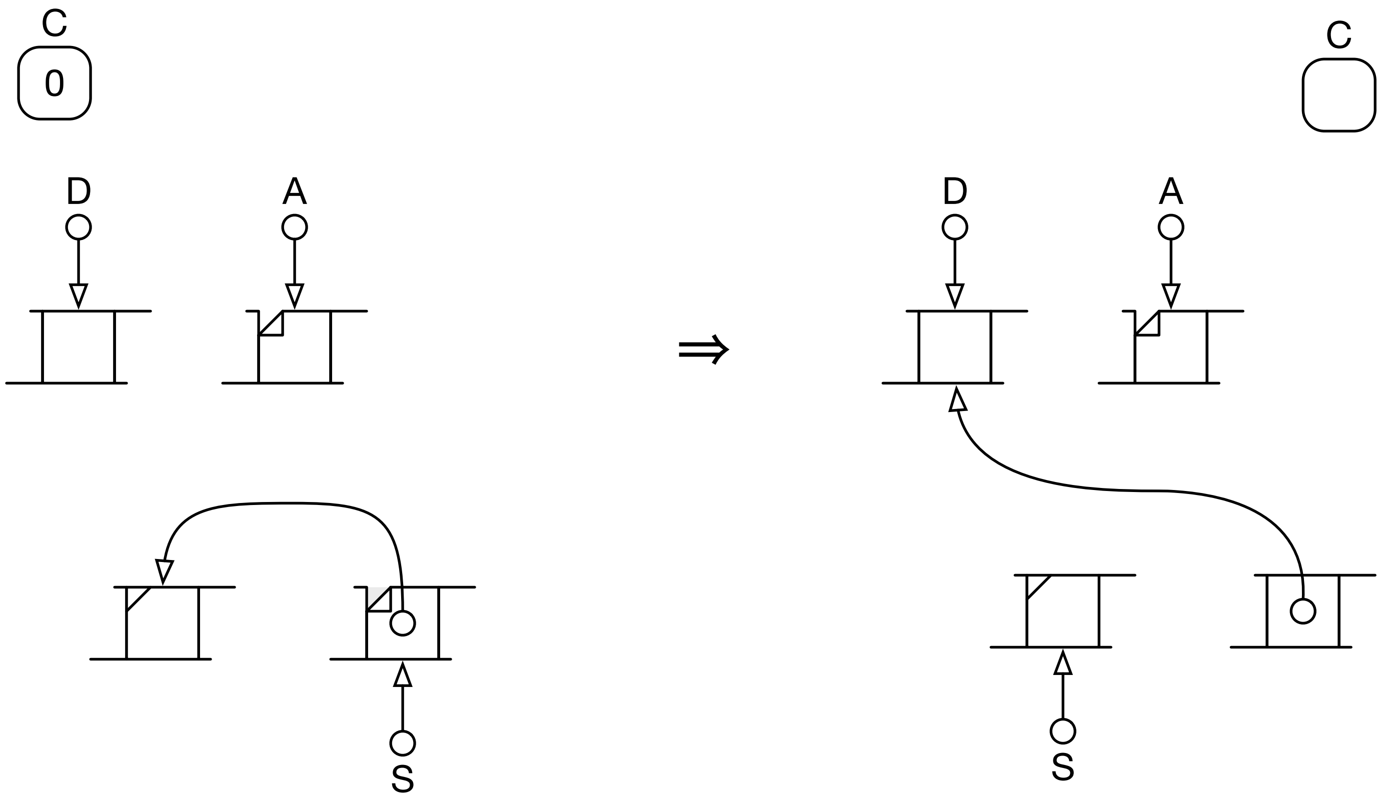
$(A = \emptyset) \wedge (S\downarrow = \mathcal{u})$



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

Pointer to an initial unmarked header cell

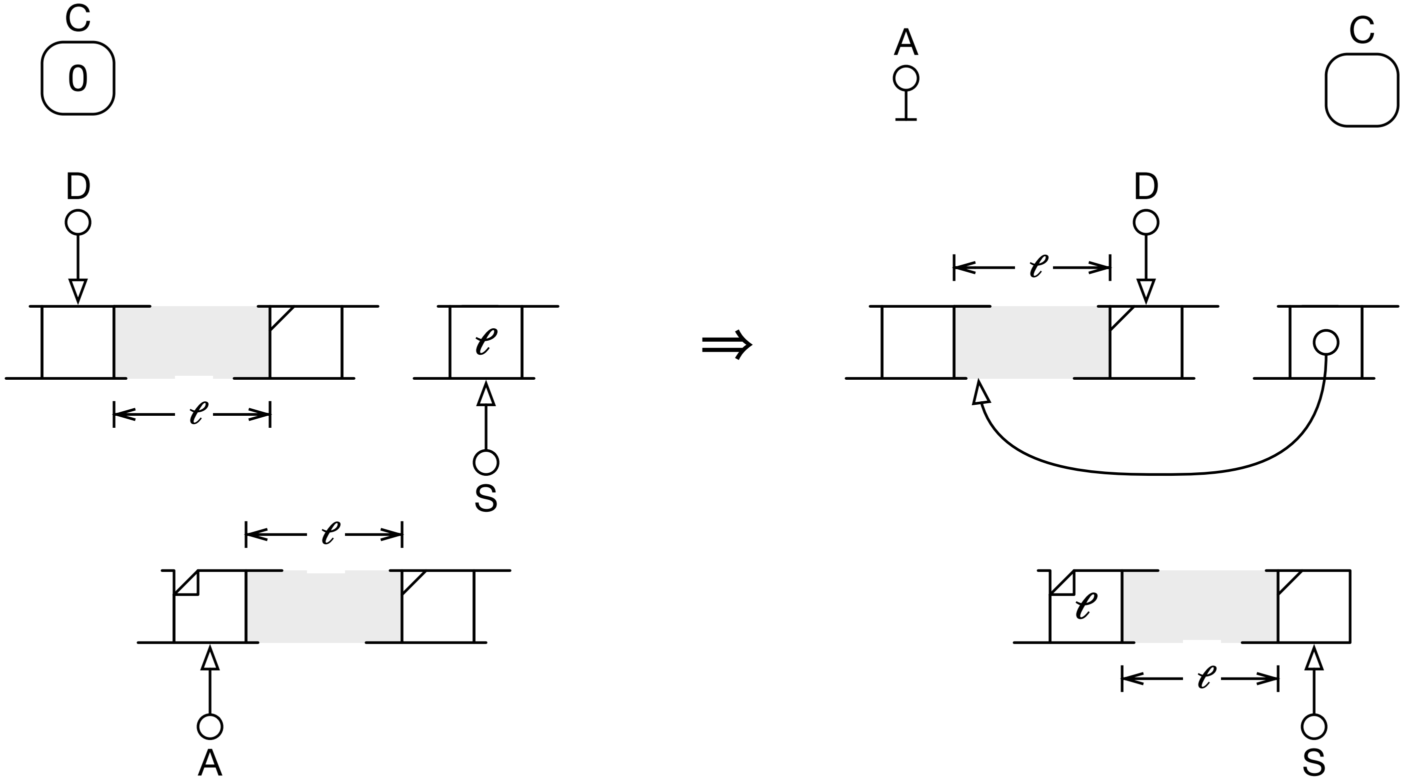
$(A \neq \emptyset) \wedge (C = 0) \wedge (S\downarrow = m)$



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

Pointer to a marked thread cell

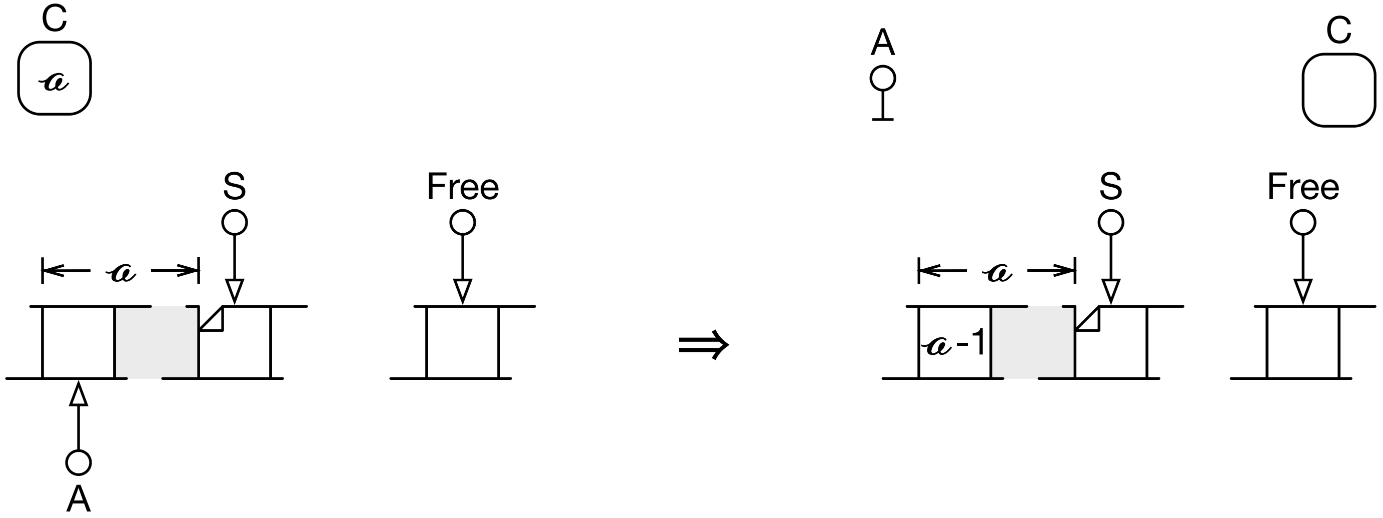
$(A \neq \varnothing) \wedge (C = 0) \wedge (S{\downarrow} = u)$



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

Pointer to an unmarked thread cell

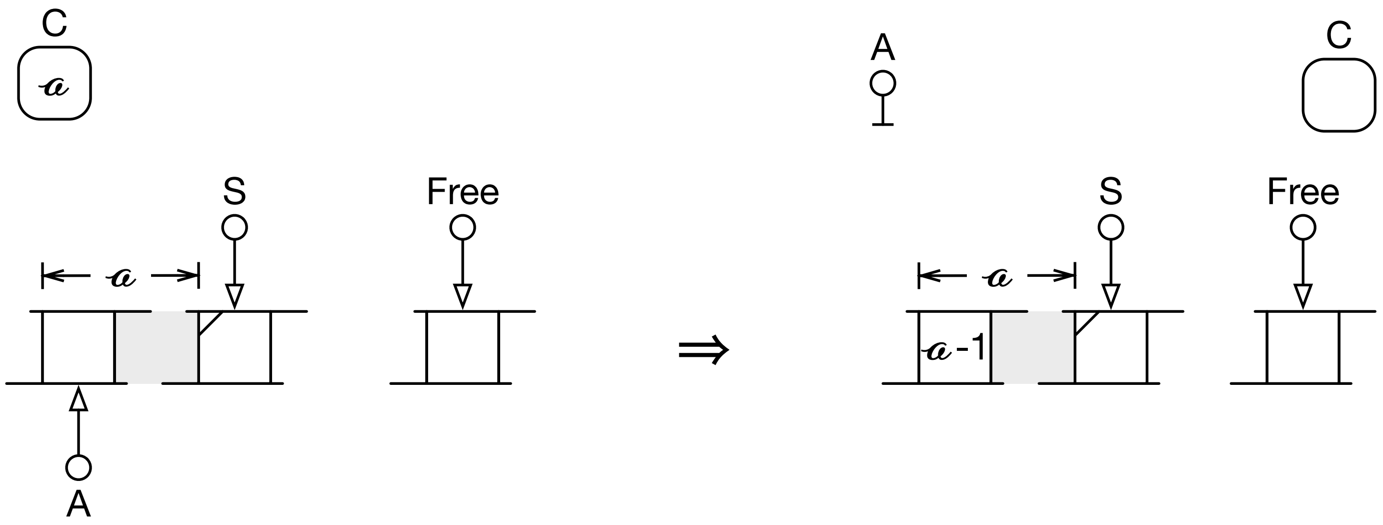
$(A \neq \varnothing) \land (C > 0) \land (S{\downarrow} = m)$



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

Pointer to a marked cell while accumulating unmarked chunks

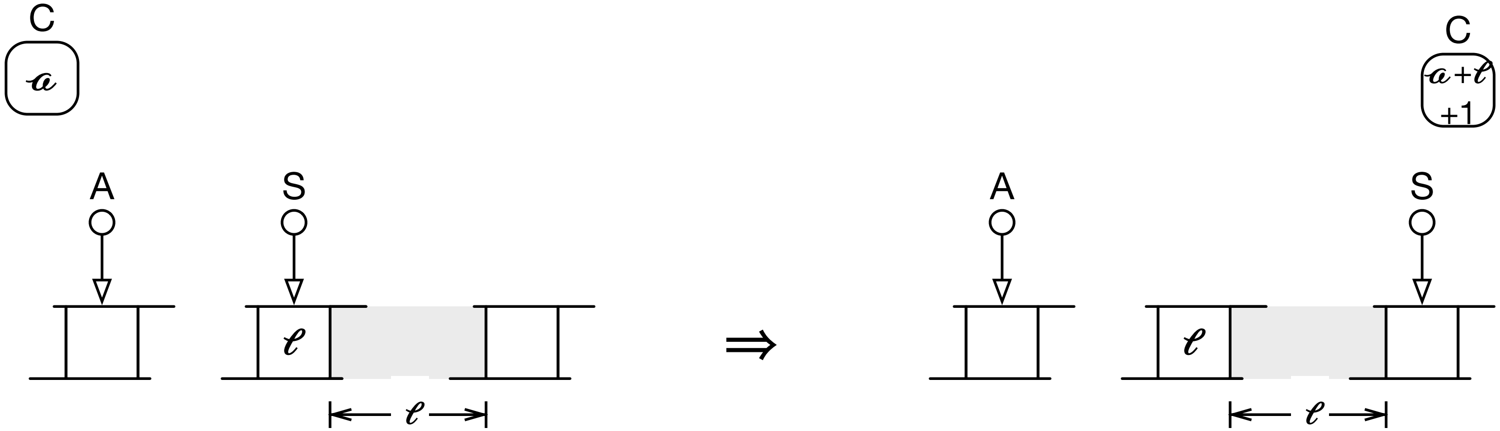
$(A \neq \varnothing) \land (C > 0) \land (S{\downarrow} = \mathcal{u}) \land (C + \mathrm{size}(S{\uparrow}) > \max)$



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

Overflow while accumulating unmarked chunks

$(A \neq \emptyset) \wedge (C > 0) \wedge (C + \mathrm{size}(S\uparrow) \leq \mathrm{max})$



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

Pointer to a consecutive unmarked header cell

```
typedef struct CEL * ptr;
typedef enum {a, m, u} flg;
typedef struct CEL { ptr P; flg F; } cel;

const ptr Null;

ptr Memory, Free;

unsigned size(ptr);
ptr stretch(unsigned);

void Jonkers_unthread(void)
  { ptr A, D, S, S_;
    unsigned C, L;
    A = Null;                          // A <- Null
    *Free = (cel){ Null, m };          // *Free = [Null, m]
    for (S = D = Memory + 1;;)         // S <- D <- Memory + 1
      { S_ = S->P;                     // S^
        if (A == Null)                 // A = Null
          if (S->F == m)               // Sv = m
            if (S < Free)              // S < Free
              { A = S;                 // A <- S
                S = S_;                // S <- S^
                C = 0; }               // C <- 0
            else                       // S = Free
              break;                   // stop
          else                         // Sv = u
            { L = size(S_);            // size(S^)
              A = S;                   // A <- S
              C = L + 1;               // C <- size(S^) + 1
              S += C; }                // S <- S + size(S^) + 1
```

```
else                           // A ≠ Null
  if (C == 0)                  // C = 0
    if (S->F == m)             // Sv = m
      { *S = (cel){ D, u };    // *S <- [D, u]
        S = S_; }              // S <- S^
    else                       // Sv = u
      { L = size(S_);          // size(S^)
        *A = (cel){ S_, m };   // *A <- [S^, m]
        *S = (cel){ D, u };    // *S <- [D, u]
        S = A + L + 1;         // S <- A + size(S^) + 1
        D += L + 1;            // D <- D + size(S^) + 1
        A = Null; }            // A <- Null
  else                         // C > 0
    if (S->F == m)             // Sv = m
      { A->P = stretch(C);     // A^ <- stretch(C)
        A = Null; }            // A <- Null
    else                       // Sv = u
      { L = size(S_);          // size(S^)
        if (C + L > max)       // C + size(S↑) > max
          { A->P = stretch(C); // A^ <- stretch(C)
            A = Null; }        // A <- Null
        else                   // C + size(S↑) ≤ max
          { S += L + 1;        // S <- S + size(S^) + 1
            C += L + 1; }}}}   // C <- C + size(S^) + 1
```