

Verification of Chase-Lev work-stealing deque

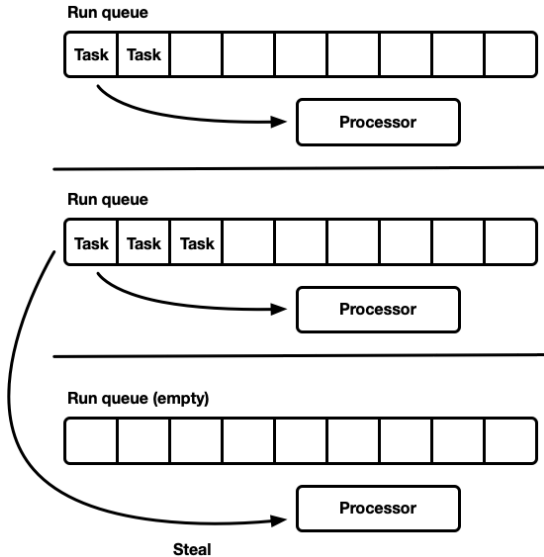
Clément Allain
François Pottier

May 5, 2023

Verification of a scheduler

```
let rec fib pool n =  
  if n < 2 then 1 else  
  let r1 = async pool (fun () -> fib_par (n - 1)) in  
  let r2 = async pool (fun () -> fib_par (n - 2)) in  
  await pool r1 + await pool r2
```

Work-stealing



Work-stealing algorithms

1. Frigo, Leiserson & Randall (1998)
 - ▶ at the core of Cilk 5
 - ▶ lock
2. Arora, Blumofe & Plaxton (2001)
 - ▶ no lock
 - ▶ one fixed size array (not circular), can overflow
3. Hendler, Lev & Shavit (2004)
 - ▶ no lock
 - ▶ list of small size arrays, no overflow
 - ▶ memory leak?
4. Chase & Lev (2005)
 - ▶ no lock
 - ▶ circular arrays, no overflow

Why is it interesting?

- ▶ demonstration of Iris on a (simplified) real-life concurrent data structure
- ▶ rich ghost state to enforce a subtle protocol
 - ▶ logical state \neq physical state
 - ▶ external future-dependent linearization point
- ▶ use of prophecy variables (with memory)

The rest of this talk

- ▶ specification using logically atomic triples
- ▶ rough idea of how the data structure works
- ▶ why we need prophecy variables (with memory)

Specification — chaselev_make

$$\frac{\{ \text{True} \}}{\text{chaselev_make } ()}$$
$$\left\{ \lambda t. \text{chaselev-inv } t \iota * \text{chaselev-model } t [] * \text{chaselev-owner } t \right\}$$

Specification — chaselev_make

$$\frac{\{ \text{True} \}}{\text{chaselev_make } ()}$$
$$\left\{ \lambda t. \text{chaselev-inv } t \iota * \text{chaselev-model } t [] * \text{chaselev-owner } t \right\}$$

enforces a protocol (using an Iris invariant)

Specification — chaselev_make

$$\frac{\{ \text{True} \}}{\text{chaselev_make } ()}$$
$$\left\{ \lambda t. \text{chaselev-inv } t \iota * \text{chaselev-model } t [] * \text{chaselev-owner } t \right\}$$

asserts the list of values that the deque (logically) contains

Specification — chaselev_make

$$\frac{\{ \text{True} \}}{\text{chaselev_make } ()}$$
$$\left\{ \lambda t. \text{chaselev-inv } t \iota * \text{chaselev-model } t [] * \text{chaselev-owner } t \right\}$$

gives the owner exclusive access to his end of the deque

Specification — chaselev_push

$$\frac{\left\{ \text{chaselev-inv } t \iota * \text{chaselev-owner } t \right\}}{\frac{\left\langle \forall vs \cdot \text{chaselev-model } t \text{ } vs \right\rangle}{\frac{\text{chaselev_push } t \text{ } v, \uparrow \iota}{\left\langle \exists \cdot \text{chaselev-model } t \text{ } (vs \# [v]) \right\rangle}}}\left\{ \lambda () \cdot \text{chaselev-owner } t \right\}$$

Specification — chaselev_push

Specification of a concurrent operation (\simeq transaction):
standard triple + logically atomic triple

$$\frac{\frac{\frac{\{P\}}{\langle \forall \overline{x} \cdot P_{\text{lin}} \rangle}}{e, \mathcal{E}}}{\langle \exists \overline{y} \cdot Q_{\text{lin}} \rangle} \frac{}{\{\lambda res \cdot Q\}}$$

P : private precondition

Q : private postcondition

P_{lin} : public precondition

Q_{lin} : public postcondition

Specification — chaselev_push

For a concurrent data structure:

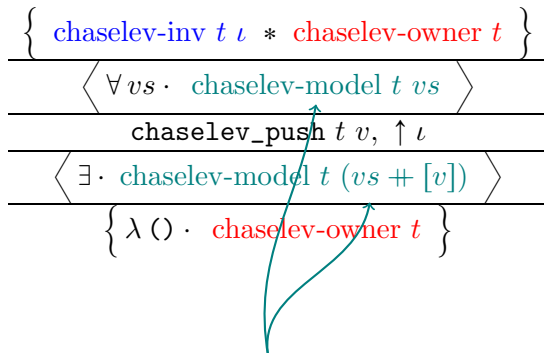
$$\frac{\frac{\frac{\{ ???\text{-inv} \cdots * P \}}{\langle \forall \overline{x} \cdot ???\text{-model} \cdots \rangle}}{e, \mathcal{E}}}{\langle \exists \overline{y} \cdot ???\text{-model} \cdots \rangle} \frac{}{\{ \lambda res \cdot Q \}}$$

Specification — chaselev_push

$$\frac{\left\{ \text{chaselev-inv } t \ \iota \ * \ \text{chaselev-owner } t \right\}}{\frac{\left\langle \forall \, vs \cdot \text{chaselev-model } t \, vs \right\rangle}{\text{chaselev_push } t \, v, \uparrow \iota} \left\langle \exists \cdot \text{chaselev-model } t \, (vs \uplus [v]) \right\rangle}$$

owner's operation

Specification — chaselev_push

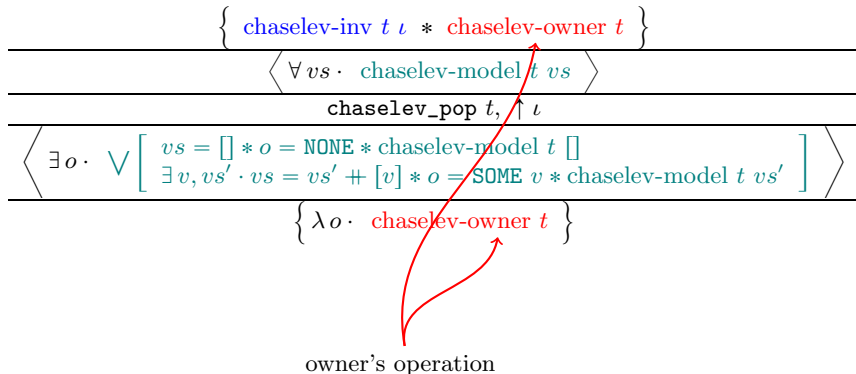


v is atomically pushed at the owner's end

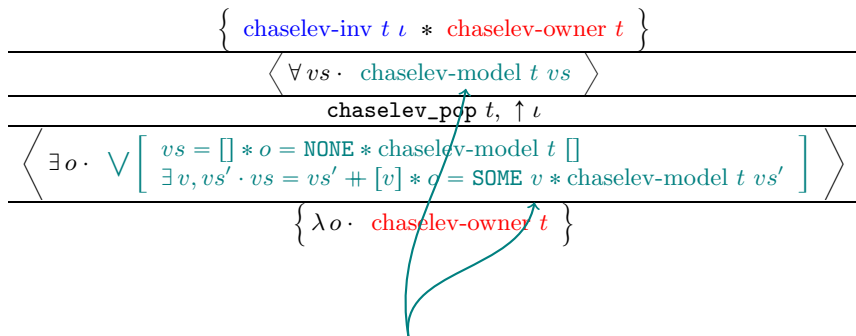
Specification — chaselev_pop

$$\frac{\left\{ \text{chaselev-inv } t \iota * \text{chaselev-owner } t \right\}}{\frac{\left\langle \forall vs \cdot \text{chaselev-model } t \text{ } vs \right\rangle}{\text{chaselev_pop } t, \uparrow \iota} \left\langle \exists o \cdot \bigvee \left[\begin{array}{l} vs = [] * o = \text{NONE} * \text{chaselev-model } t [] \\ \exists v, vs' \cdot vs = vs' \# [v] * o = \text{SOME } v * \text{chaselev-model } t \text{ } vs' \end{array} \right] \right\rangle} \left\{ \lambda o \cdot \text{chaselev-owner } t \right\}$$

Specification — chaselev_pop



Specification — chaselev_pop

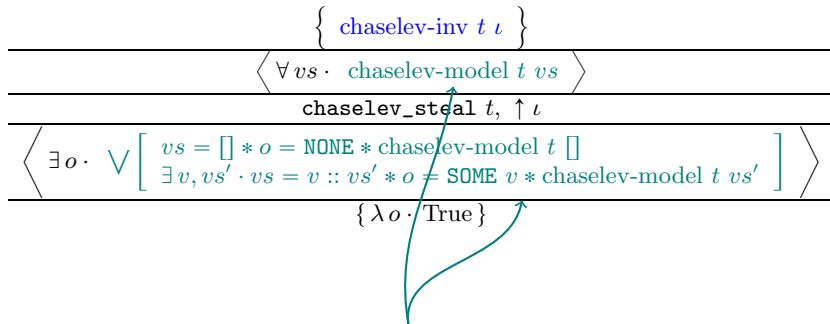


either 1) the deque is seen empty
 or 2) some value v is atomically popped at the owner's end

Specification — chaselev_steal

$$\frac{\left\{ \text{chaselev-inv } t \ \iota \right\}}{\frac{\left\langle \forall vs \cdot \text{chaselev-model } t \ vs \right\rangle}{\text{chaselev_steal } t, \uparrow \iota} \left\langle \exists o \cdot \bigvee \left[\begin{array}{l} vs = [] * o = \text{NONE} * \text{chaselev-model } t \ [] \\ \exists v, vs' \cdot vs = v :: vs' * o = \text{SOME } v * \text{chaselev-model } t \ vs' \end{array} \right] \right\rangle} \left\{ \lambda o \cdot \text{True} \right\}$$

Specification — chaselev_steal



either 1) the deque is seen empty
or 2) some value v is atomically popped at the thieves' end

Physical state



data: infinite array storing all values

Physical state



data: infinite array storing all values

front: *monotone* index for thieves' end

Physical state

data {

CHASELEVFRONTLBGET

$\gamma.\text{front}$
 $\bullet \text{ front}$

$\gamma.\text{front}$
 $\circ \text{ front}$

data: in
 front: m

CHASELEVFRONTVALID

$\gamma.\text{front}$ $\gamma.\text{front}$
 $\bullet \text{ front}_1$ $\circ \text{ front}_2$

$\text{front}_2 \leq \text{front}_1$

CHASELEVFRONTUPDATE

$\text{front} \leq \text{front}'$ $\gamma.\text{front}$
 $\bullet \text{ front}$

$\gamma.\text{front}$
 $\bullet \text{ front}'$

Physical state

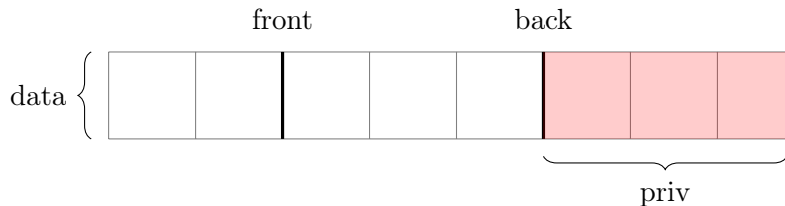


data: infinite array storing all values

front: *monotone* index for thieves' end

back: index for owner's end

Physical state



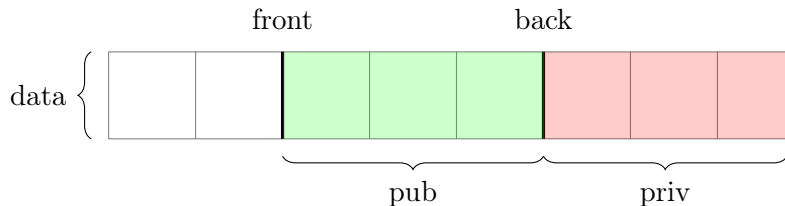
data: infinite array storing all values

front: *monotone* index for thieves' end

back: index for owner's end

priv: list of private values (controlled by owner)

Physical state



data: infinite array storing all values

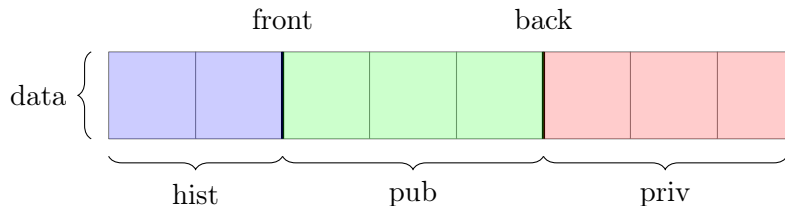
front: *monotone* index for thieves' end

back: index for owner's end

priv: list of private values (controlled by owner)

pub: list of public values (= model)

Physical state



data: infinite array storing all values

front: *monotone* index for thieves' end

back: index for owner's end

priv: list of private values (controlled by owner)

pub: list of public values (= model)

hist: *monotone* list of history values

Physical state

data {

CHASELEVHISTLBGET

$\boxed{\bullet \text{ } hist}_{\gamma.hist}$

$\boxed{\circ \text{ } hist}_{\gamma.hist}$

CHASELEVHISTVALID

$\boxed{\bullet \text{ } hist_1}_{\gamma.hist} \quad \boxed{\circ \text{ } hist_2}_{\gamma.hist}$

$hist_2 \sqsubseteq_{\text{prefix}} hist_1$

CHASELEVHISTUPDATE

$\boxed{\bullet \text{ } hist}_{\gamma.hist}$

$\boxed{\bullet (hist \uplus [v])}_{\gamma.hist}$

data: in

front: m

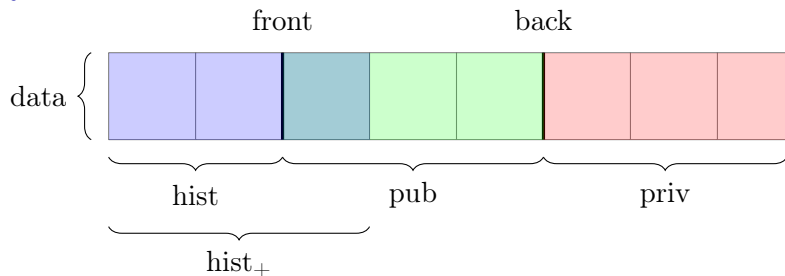
back: in

priv: lis

pub: lis

hist: m

Physical state



data: infinite array storing all values

front: *monotone* index for thieves' end

back: index for owner's end

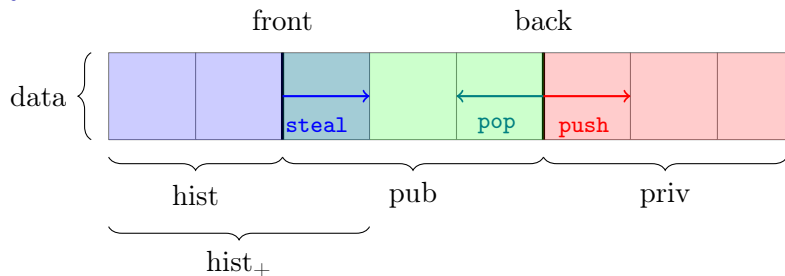
priv: list of private values (controlled by owner)

pub: list of public values (= model)

hist: *monotone* list of history values

hist₊: *monotone* list of extended history values

Physical state



data: infinite array storing all values

front: *monotone* index for thieves' end

back: index for owner's end

priv: list of private values (controlled by owner)

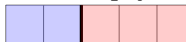
pub: list of public values (= model)

hist: *monotone* list of history values

hist₊: *monotone* list of extended history values

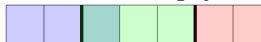
Logical state

① empty



front = back

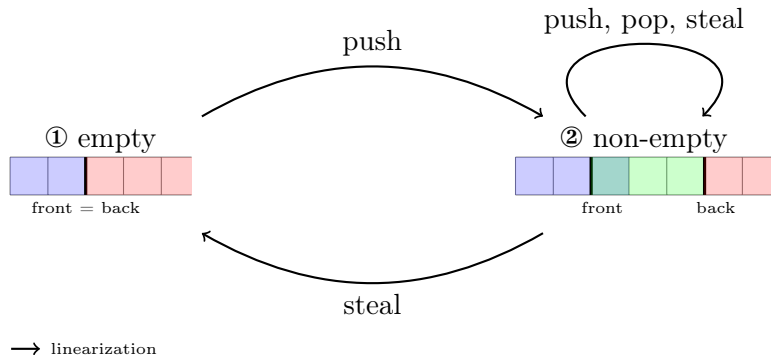
② non-empty



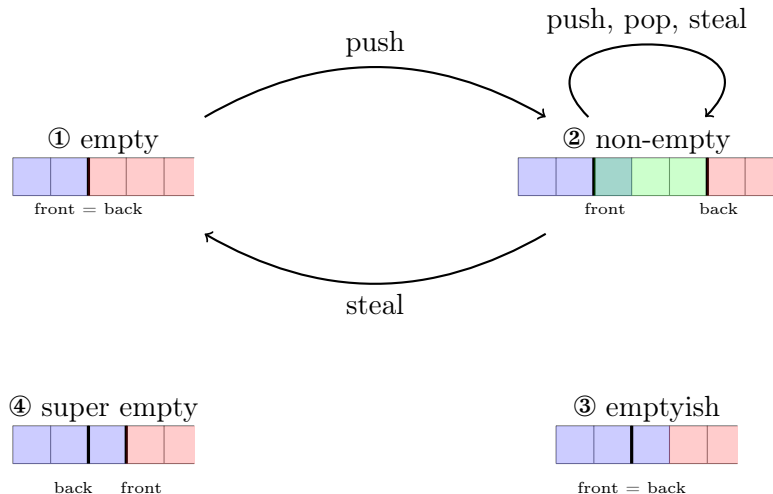
front

back

Logical state

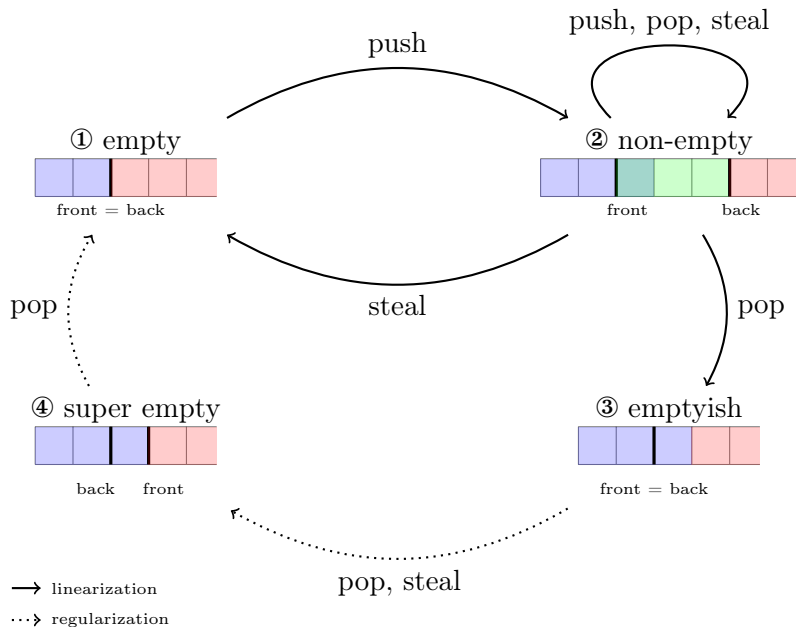


Logical state

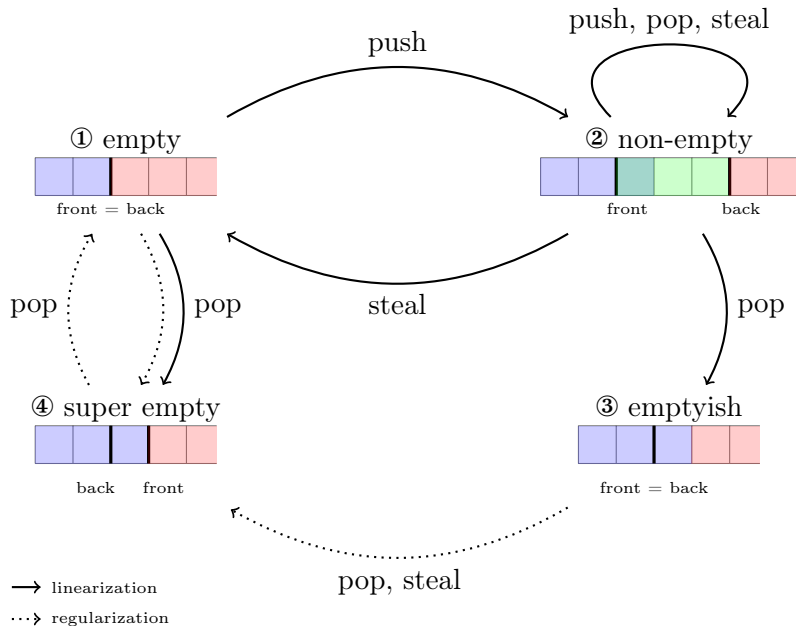


→ linearization

Logical state



Logical state



Prophecy variable

$\{ \text{True} \} \text{ NewProph } \{ \lambda p \cdot \exists \text{prophs} \cdot \text{proph } p \text{ prophs} \}$

$$\frac{\text{atomic } e \quad \text{proph } p \text{ prophs} \quad \left\{ \begin{array}{l} \lambda w \cdot \forall \text{prophs}' \cdot \\ \text{prophs} = (w, v) :: \text{prophs}' \text{ } -* \\ \text{proph } p \text{ prophs}' \text{ } -* \\ \Phi \ w \end{array} \right\}}{\text{WP Resolve } e \ p \ v \ \{ \Phi \}}$$

Back to *The Future is Ours* (Jung *et al.*)

```
let rdcss rm rn m1 n1 n2 =  
  let p = NewProph in  
  let descr = ref (rm, m1, n1, n2, p) in  
  ...  
  
let complete descr rn =  
  let (rm, m1, n1, n2, p) = !descr in  
  let id = NewId in  
  let m = !rm in  
  let n_new = if m = m1 then n2 else n1 in  
  Resolve (CmpXchg rn (inr descr) (inl n_new)) p id ;  
  ()
```

Prophecy variable with memory

$\{\text{True}\} \text{ NewProph } \{ \lambda p \cdot \exists \gamma, \text{prophs} \cdot \text{proph } p \ \gamma \sqcap \text{prophs} \}$

$$\frac{\text{atomic } e \quad \text{proph } p \ \gamma \text{ past prophs} \quad \left\{ \begin{array}{l} \lambda w \cdot \forall \text{prophs}' \cdot \\ \text{prophs} = (w, v) :: \text{prophs}' \rightarrow * \\ \text{proph } p \ \gamma (\text{past} \# [(w, v)]) \text{ prophs}' \rightarrow * \\ \Phi \ w \end{array} \right\}}{\text{WP Resolve } e \ p \ v \ \{ \Phi \}}$$

Prophecy variable with memory

$$\frac{\text{PROPHECYLBGET} \quad \text{proph } p \ \gamma \ \text{past } \textcolor{blue}{prophs}}{\text{proph-lb } \gamma \ \textcolor{blue}{prophs}}$$

$$\frac{\text{PROPHECYVALID} \quad \text{proph } p \ \gamma \ \textcolor{red}{past} \ \textcolor{blue}{prophs}_1 \quad \text{proph-lb } \gamma \ \textcolor{teal}{prophs}_2}{\exists \textcolor{red}{past}_1, \textcolor{orange}{past}_2 \cdot \bigwedge \left[\begin{array}{l} \textcolor{red}{past} = \textcolor{red}{past}_1 \# \textcolor{orange}{past}_2 \\ \textcolor{orange}{past}_2 \ \# \textcolor{blue}{prophs}_1 = \textcolor{teal}{prophs}_2 \end{array} \right]}$$

Thank you for your attention!

Implementation — chaselev_make

```
let chaselev_make _ =  
  let t = AllocN 4 () in  
  t.front <- 0 ;  
  t.back <- 0 ;  
  t.data <- inf_array_make () ;  
  t.prophecy <- NewProph ;  
  t
```

Implementation — chaselev_push

```
let chaselev_push t v =  
  let back = !t.back in  
  inf_array_set !t.data back v ;  
  t.back <- back + 1
```

Implementation — chaselev_steal

```
let rec chaselev_steal t =  
  let id = NewId in  
  let front = !t.front in  
  let back = !t.back in  
  if front < back then (  
    if Snd (  
      Resolve (  
        CmpXchg t.front front (front + 1)  
      ) !t.prophecy (front, id)  
    ) then (  
      SOME (inf_array_get !t.data front)  
    ) else (  
      chaselev_steal t  
    )  
  ) else (  
    NONE  
  )
```

Implementation — chaselev_pop

```
let chaselev_pop t =  
  let id = NewId in  
  let back = !t.back - 1 in  
  t.back <- back ;  
  let front = !t.front in  
  if back < front then (  
    t.back <- front  
  ) else (  
    if front < back then (  
      SOME (inf_array_get !t.data back)  
    ) else (  
      if Snd (  
        Resolve (  
          CmpXchg t.front front (front + 1)  
        ) !t.prophecy (front, id)  
      ) then (  
        t.back <- front + 1 ;  
        SOME (inf_array_get !t.data back)  
      ) else (  
        t.back <- front + 1 ;  
        NONE  
      )  
    )  
  )
```

Infinite array

$$\frac{\{ \text{True} \}}{\text{inf_array_make } v}$$
$$\frac{}{\{ \lambda \text{ arr} \cdot \text{inf-array-model } \text{arr} \ (\lambda_ \cdot v) \}}$$

$$\frac{\langle \forall \text{ vs} \cdot \text{inf-array-model } \text{arr} \ \text{vs} * 0 \leq i \rangle}{\text{inf_array_get } \text{arr} \ i}$$
$$\frac{}{\langle \exists \cdot \lambda (\text{vs} i) \cdot \text{inf-array-model } \text{arr} \ \text{vs} \rangle}$$

$$\frac{\langle \forall \text{ vs} \cdot \text{inf-array-model } \text{arr} \ \text{vs} * 0 \leq i \rangle}{\text{inf_array_set } \text{arr} \ i \ v}$$
$$\frac{}{\langle \exists \cdot \lambda_ \cdot \text{inf-array-model } \text{arr} \ \text{vs} [i \mapsto v] \rangle}$$

Invariant

$$\begin{aligned} \text{chaselev-inv } t \ \iota &\triangleq \\ \exists \ell, \gamma, data, p. & \\ * \left[\begin{array}{l} t = \ell * \text{meta } \ell \ \gamma \\ \ell.\text{data} \mapsto_{\square} data * \ell.\text{prophecy} \mapsto_{\square} p \\ \boxed{\text{chaselev-inv-inner } \ell \ \gamma \ \iota \ data \ p} \end{array} \right] \end{aligned}$$

Invariant

$\text{chaselev-inv-inner } \ell \ \gamma \ \iota \ \text{data } p \stackrel{\Delta}{=}$

$\exists \text{ front, back, hist, pub, priv, past, prophs} \cdot$

$$\begin{array}{l}
 \left[\begin{array}{l}
 \ell.\text{front} \mapsto \text{front} * \ell.\text{back} \mapsto \text{back} \\
 \boxed{\bullet (back, priv)}^{\gamma.\text{ctl}} \\
 \boxed{\bullet front}^{\gamma.\text{front}} \\
 \text{inf-array-model data (hist} \uplus \text{pub) priv} \\
 \boxed{\bullet pub}^{\gamma.\text{pub}} * |pub| = (back - front)_+ \\
 \text{wise-prophet-model } p \ \gamma.\text{prophet past prophs} \\
 \forall (front', _) \in \text{past} \cdot front' < front \\
 \text{chaselev-state } \gamma \ \iota \ \text{front back hist pub prophs}
 \end{array} \right.
 \end{array}$$

State

$$\text{chaselev-state } \gamma \ \iota \ \textit{front back hist pub prophs} \triangleq \\ \bigvee \left[\begin{array}{l} \text{chaselev-state}_1 \ \gamma \ \textit{front back hist} \\ \text{chaselev-state}_2 \ \gamma \ \iota \ \textit{front back hist pub prophs} \\ \text{chaselev-lock } \gamma * \bigvee \left[\begin{array}{l} \text{chaselev-state}_3 \ \gamma \ \textit{front back hist prophs} \\ \text{chaselev-state}_4 \ \gamma \ \textit{front back hist} \end{array} \right] \end{array} \right]$$

State 1 (empty)

$$\text{chaselev-state}_1 \ \gamma \ front \ back \ hist \triangleq$$

$$* \left[\begin{array}{l} front = back \\ \begin{array}{|c|} \hline \bullet \text{ hist} \\ \hline \end{array} \quad * |hist| = front \\ \begin{array}{|c|} \hline \bullet \text{ --- } \cdot \text{ } \circ \text{ --- } \\ \hline \end{array} \end{array} \right] \gamma.\text{winner}$$

State 2 (non-empty)

$\text{chaselev-state}_2 \gamma \iota \text{ front back hist pub prophs} \triangleq$

$$\begin{array}{l}
 \left[\begin{array}{l}
 \text{front} < \text{back} \\
 \boxed{\bullet (\text{hist} \# [\text{pub}[0]])}^{\gamma.\text{hist}} * |\text{hist}| = \text{front} \\
 \\
 \mathbf{match} \text{ filter } (\lambda(\text{front}', _) \cdot \text{front}' = \text{front}) \text{ prophs } \mathbf{with} \\
 | [] \Rightarrow \boxed{\bullet - \circ -}^{\gamma.\text{winner}} \\
 | (_, id) :: _ \Rightarrow \\
 \bigvee \left[\begin{array}{l}
 \boxed{\bullet - \circ -}^{\gamma.\text{winner}} \\
 \text{identifier } id * \exists \Phi \cdot \boxed{\bullet (\text{front}, \Phi)}^{\gamma.\text{winner}} * \text{chaselev-au } \gamma \iota \Phi
 \end{array} \right.
 \end{array} \right]
 \end{array}$$

State 3 (emptyish)

$$\text{chaselev-state}_3 \gamma \text{ front back hist prophs} \triangleq$$

$$* \left[\begin{array}{l} \text{front} = \text{back} \\ \boxed{\bullet \text{ hist}}^{\gamma.\text{hist}} * |\text{hist}| = \text{front} + 1 \\ \\ \text{match filter } (\lambda(\text{front}', _) \cdot \text{front}' = \text{front}) \text{ prophs with} \\ | \square \Rightarrow \boxed{\circ(\text{front}, -)}^{\gamma.\text{winner}} \\ | _ \Rightarrow \exists \Phi \cdot \boxed{\bullet(\text{front}, \Phi)}^{\gamma.\text{winner}} * \Phi(\text{SOME hist}[\text{front}]) \end{array} \right.$$

State 4 (super empty)

$$\text{chaselev-state}_4 \gamma \text{ front back hist} \triangleq$$

$$* \left[\begin{array}{l} \text{front} = \text{back} + 1 \\ \begin{array}{|c|} \hline \bullet \text{ hist} \\ \hline \end{array} * |\text{hist}| = \text{front} \\ \begin{array}{|c|} \hline \bullet - \cdot \circ - \\ \hline \end{array} \end{array} \right] \gamma^{\text{hist}} \gamma^{\text{winner}}$$