# Verification of Chase-Lev work-stealing deque
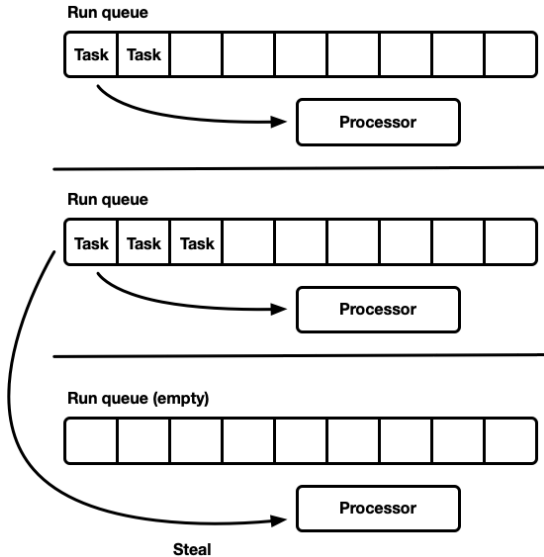
Clément Allain
François Pottier

April 25, 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 (typed) 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`

$$\frac{\{\,\text{True}\,\}}{\texttt{chaselev\_make ()}}$$
$$\left\{\,\lambda\ t\cdot\ \text{chaselev-inv}\ t\ \iota\ *\ \text{chaselev-model}\ t\ [\,]\ *\ \text{chaselev-owner}\ t\,\right\}$$

# Specification — `chaselev_make`

$$\frac{\{\text{True}\}}{\texttt{chaselev\_make ()}}$$
$$\left\{\lambda\, t \cdot\ \text{chaselev-inv}\ t\ \iota\ *\ \text{chaselev-model}\ t\ []\ *\ \text{chaselev-owner}\ t\right\}$$

enforces a protocol (using an Iris invariant)

# Specification — `chaselev_make`

$$\cfrac{\{\,\text{True}\,\}}{\texttt{chaselev\_make ()}}$$
$$\{\,\lambda\,t\cdot\ \text{chaselev-inv}\ t\ \iota\ *\ \text{chaselev-model}\ t\ []\ *\ \text{chaselev-owner}\ t\,\}$$

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

# Specification — `chaselev_make`

$$\frac{\{\,\text{True}\,\}}{\texttt{chaselev\_make ()}}$$
$$\left\{\,\lambda\,t\cdot\ \text{chaselev-inv } t\ \iota\ *\ \text{chaselev-model } t\ [\,]\ *\ \text{chaselev-owner } t\,\right\}$$

gives the owner exclusive access to his end of the deque

# Specification — `chaselev_push`

$$\left\{ \text{chaselev-inv } t\ \iota\ *\ \text{chaselev-owner } t \right\}$$

$$\left\langle \forall\, vs\, \cdot\ \text{chaselev-model } t\ vs \right\rangle$$

$$\texttt{chaselev\_push } t\ v,\ \uparrow \iota$$

$$\left\langle \exists\, \cdot\ \text{chaselev-model } t\ (vs + [v]) \right\rangle$$

$$\left\{ \lambda\, ()\, \cdot\ \text{chaselev-owner } t \right\}$$

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

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

$P$ : private precondition

$Q$ : private postcondition

$P_{\mathrm{lin}}$ : public precondition

$Q_{\mathrm{lin}}$ : public postcondition

For a concurrent data structure:

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

# Specification — `chaselev_push`

$$\left\{ \text{chaselev-inv } t\ \iota\ *\ \text{chaselev-owner } t \right\}$$

$$\left\langle \forall\, vs \cdot\ \text{chaselev-model } t\ vs \right\rangle$$

$$\texttt{chaselev\_push } t\ v,\ \uparrow \iota$$

$$\left\langle \exists \cdot\ \text{chaselev-model } t\ (vs + [v]) \right\rangle$$

$$\left\{ \lambda\,()\cdot\ \text{chaselev-owner } t \right\}$$

owner's operation

# Specification — `chaselev_push`

$$\left\{ \text{chaselev-inv } t \; \iota \; * \; \text{chaselev-owner } t \right\}$$

$$\left\langle \forall \, vs \cdot \; \text{chaselev-model } t \; vs \right\rangle$$

$$\texttt{chaselev\_push } t \; v, \; \uparrow \iota$$

$$\left\langle \exists \cdot \; \text{chaselev-model } t \; (vs + [v]) \right\rangle$$
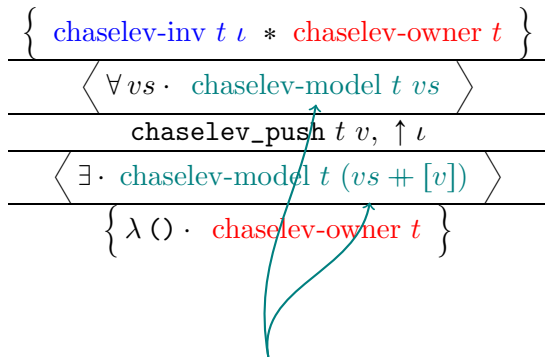
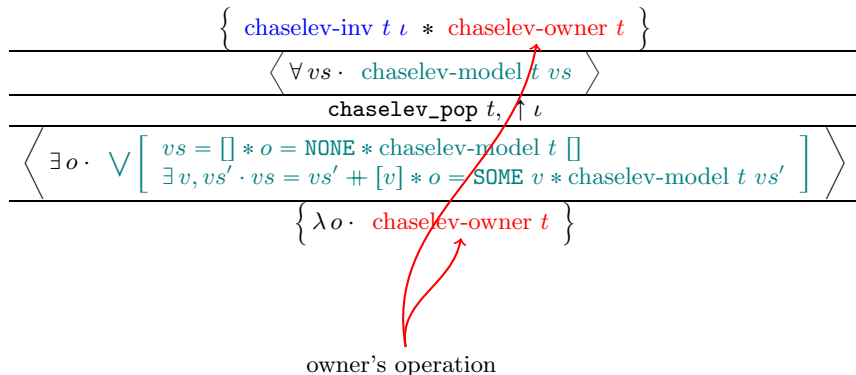$$\left\{ \lambda \, () \cdot \; \text{chaselev-owner } t \right\}$$
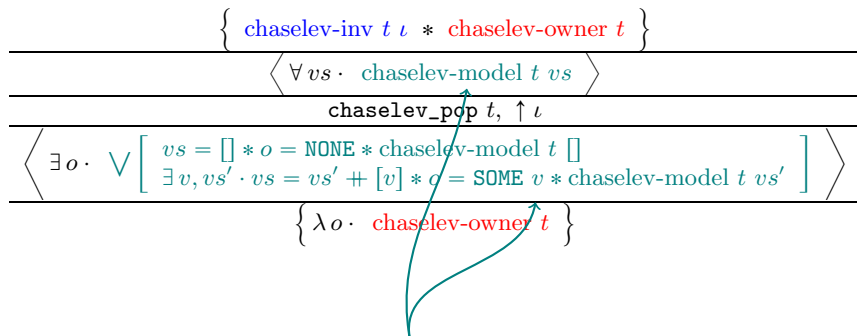
$v$ is atomically pushed at the owner's end

# Specification — `chaselev_pop`

$$\left\{ \text{chaselev-inv } t\ \iota\ *\ \text{chaselev-owner } t \right\}$$

$$\left\langle\, \forall\, vs \cdot\ \text{chaselev-model } t\ vs \,\right\rangle$$

$$\texttt{chaselev\_pop } t,\ \uparrow \iota$$

$$\left\langle\, \exists\, o \cdot\ \bigvee \left[ \begin{array}{l} vs = [\,] * o = \texttt{NONE} * \text{chaselev-model } t\ [\,] \\ \exists\, v, vs' \cdot vs = vs' +\!\!+ [v] * o = \texttt{SOME } v * \text{chaselev-model } t\ vs' \end{array} \right] \,\right\rangle$$

$$\left\{ \lambda\, o \cdot\ \text{chaselev-owner } t \right\}$$

# Specification — `chaselev_pop`



$$\left\{ \text{chaselev-inv } t \; \iota \; * \; \text{chaselev-owner } t \right\}$$

$$\left\langle \; \forall \, vs \cdot \; \text{chaselev-model } t \; vs \; \right\rangle$$

$$\texttt{chaselev\_pop } t, \; \iota$$

$$\left\langle \; \exists \, o \cdot \; \bigvee \left[ \begin{array}{l} vs = [] * o = \texttt{NONE} * \text{chaselev-model } t \; [] \\ \exists \, v, vs' \cdot vs = vs' + [v] * o = \texttt{SOME } v * \text{chaselev-model } t \; vs' \end{array} \right] \; \right\rangle$$

$$\left\{ \lambda \, o \cdot \; \text{chaselev-owner } t \right\}$$

owner's operation

# Specification — `chaselev_pop`

$$\left\{ \text{chaselev-inv } t \; \iota \; * \; \text{chaselev-owner } t \right\}$$

$$\left\langle \forall vs \cdot \; \text{chaselev-model } t \; vs \right\rangle$$

$$\texttt{chaselev\_pop } t, \; \uparrow \iota$$

$$\left\langle \exists o \cdot \; \bigvee \left[ \begin{array}{l} vs = [] * o = \texttt{NONE} * \text{chaselev-model } t \; [] \\ \exists v, vs' \cdot vs = vs' + [v] * o = \texttt{SOME } v * \text{chaselev-model } t \; vs' \end{array} \right] \right\rangle$$

$$\left\{ \lambda o \cdot \; \text{chaselev-owner } t \right\}$$

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

# Specification — `chaselev_steal`

$$\frac{\left\{ \text{ chaselev-inv } t\ \iota \ \right\}}{\left\langle \forall\, vs \cdot \ \text{ chaselev-model } t\ vs \right\rangle}$$

$$\texttt{chaselev\_steal } t,\ \uparrow \iota$$

$$\left\langle \exists\, o \cdot\ \bigvee \left[ \begin{array}{l} vs = [\,] * o = \texttt{NONE} * \text{chaselev-model } t\ [\,] \\ \exists\, v, vs' \cdot vs = v :: vs' * o = \texttt{SOME } v * \text{chaselev-model } t\ vs' \end{array} \right] \right\rangle$$

$$\{\, \lambda\, o \cdot \text{True} \,\}$$

# Specification — `chaselev_steal`

$$\left\{ \; \text{chaselev-inv } t \; \iota \; \right\}$$

$$\left\langle \; \forall \, vs \cdot \;\; \text{chaselev-model } t \; vs \; \right\rangle$$

$$\texttt{chaselev\_steal } t, \; \uparrow \iota$$

$$\left\langle \; \exists \, o \cdot \; \bigvee \left[ \begin{array}{l} vs = [\,] * o = \texttt{NONE} * \text{chaselev-model } t \; [\,] \\ \exists \, v, vs' \cdot vs = v :: vs' * o = \texttt{SOME } v * \text{chaselev-model } t \; vs' \end{array} \right] \; \right\rangle$$

$$\{ \, \lambda \, o \cdot \text{True} \, \}$$
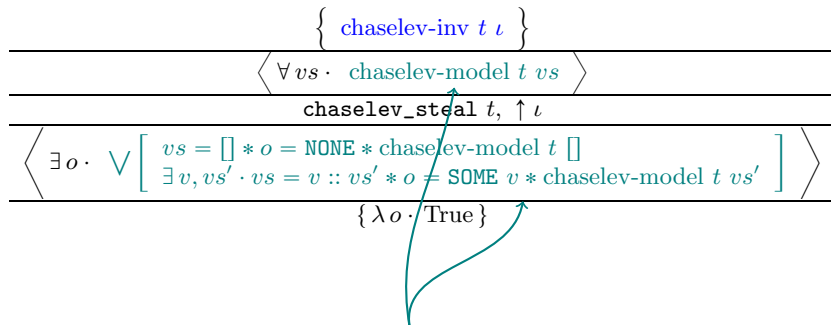
either 1) the deque is seen empty

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 $\Big\{$

data: in

front: $m$

$$\text{CHASELEVFRONTVALID}$$

$$\boxed{\bullet\ front_1}^{\gamma.\text{front}} \qquad \boxed{\circ\ front_2}^{\gamma.\text{front}}$$

$$\overline{\qquad\qquad front2 \leqslant front1 \qquad\qquad}$$

$$\text{CHASELEVFRONTUPDATE}$$

$$front \leqslant front' \qquad \boxed{\bullet\ front}^{\gamma.\text{front}}$$

$$\overline{\qquad\qquad \boxed{\bullet\ front'}^{\gamma.\text{front}} \qquad\qquad}$$

$$\text{CHASELEVFRONTFRAGGET}$$

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

$$\overline{\qquad\qquad}$$

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

# Physical state



data: infinite array storing all values

front: *monotone* index for thieves' end

back: index for owner's end

# Physical state
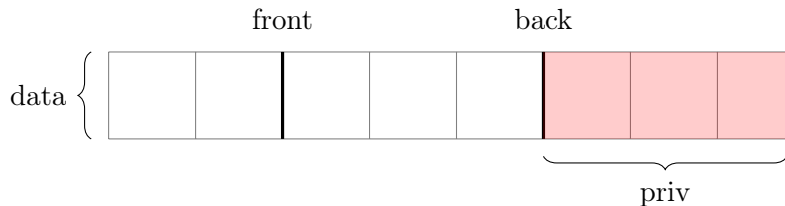


data: infinite array storing all values

front: *monotone* index for thieves' end

back: index for owner's end
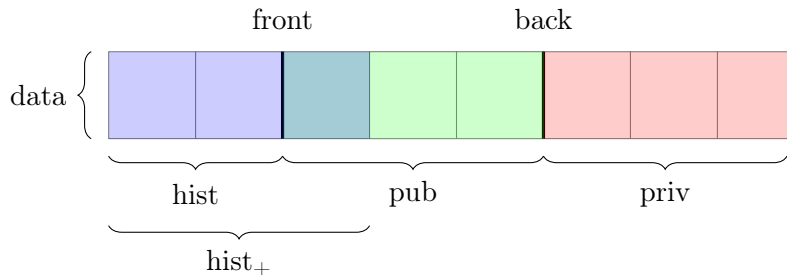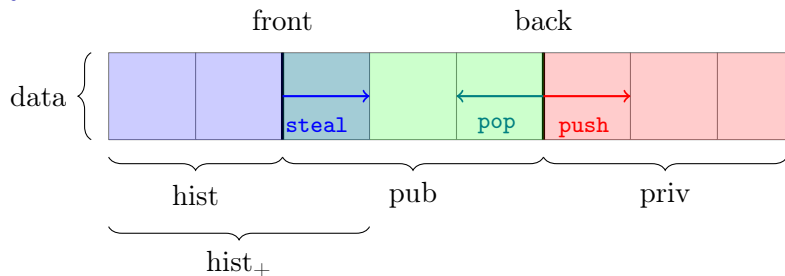
priv: list of private values (controlled by owner)

# Physical state
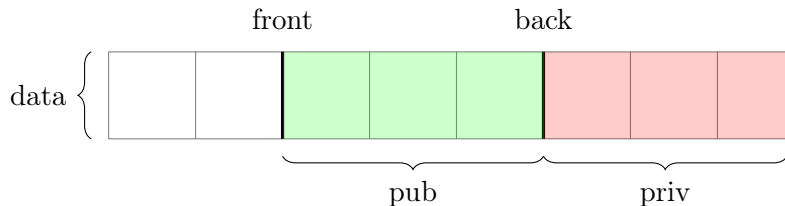


data: infinite array storing all values

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: in
front: *m*
back: in

priv: lis
pub: lis
hist: *monotone* list of 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

# 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
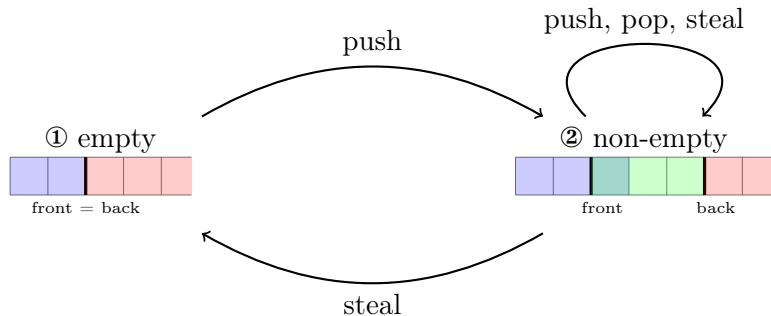
# Logical state

① empty
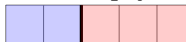


front = back
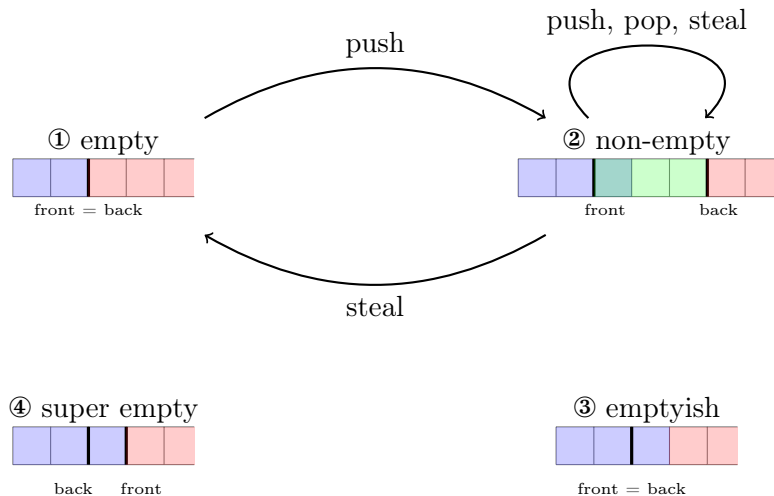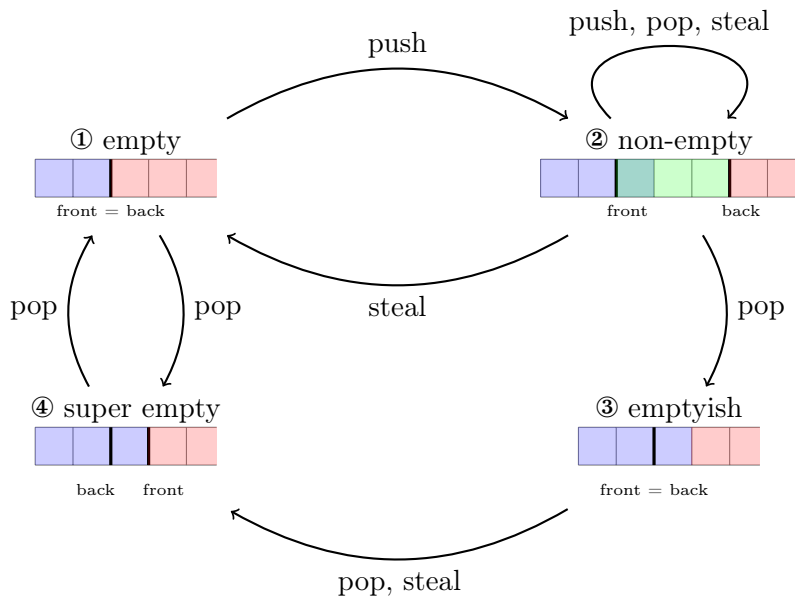
② non-empty
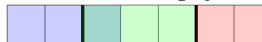


front          back

# Logical state

# Logical state

# Logical state

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
      )
```

## Invariant

$$\text{chaselev-inv } t \; \iota \triangleq$$

$$\exists \, \ell, \gamma, \mathit{data}, p \cdot$$

$$* \left[ \begin{array}{l} t = \ell * \text{meta } \ell \; \gamma \\ \ell.\text{data} \mapsto_\square \mathit{data} * \ell.\text{prophecy} \mapsto_\square p \\ \boxed{\text{chaselev-inv-inner } \ell \; \gamma \; \iota \; \mathit{data} \; p} \end{array} \right]^\iota$$

## Invariant

$\text{chaselev-inv-inner } \ell \; \gamma \; \iota \; data \; p \triangleq$

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

$$* \left[ \begin{array}{l} \ell.\text{front} \mapsto front * \ell.\text{back} \mapsto back \\[2pt] \boxed{\bullet \, (back, priv)}^{\;\gamma.\text{ctl}} \\[2pt] \boxed{\bullet \, front}^{\;\gamma.\text{front}} \\[2pt] \text{inf-array-model } data \; (hist + pub) \; priv \\[2pt] \boxed{\bullet \, pub}^{\;\gamma.\text{pub}} \quad * \; |pub| = (back - front)_+ \\[2pt] \text{wise-prophet-model } p \; \gamma.\text{prophet } past \; prophs \\[2pt] \forall (front', \_) \in past \cdot front' < front \\[2pt] \text{chaselev-state } \gamma \; \iota \; front \; back \; hist \; pub \; prophs \end{array} \right.$$

# State

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

$$\bigvee \left[ \begin{array}{l} \text{chaselev-state}_1 \ \gamma \ front \ back \ hist \\ \text{chaselev-state}_2 \ \gamma \ \iota \ front \ back \ hist \ pub \ prophs \\ \text{chaselev-lock } \gamma * \bigvee \left[ \begin{array}{l} \text{chaselev-state}_{3,1} \ \gamma \ front \ back \ hist \ prophs \\ \text{chaselev-state}_{3,2} \ \gamma \ front \ back \ hist \end{array} \right. \end{array} \right.$$

# State

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

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

# State

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

$$
* \left[
\begin{array}{l}
front < back \\[4pt]
\boxed{\bullet \, (hist +\!\!+ [pub[0]])}^{\gamma.\text{hist}} \quad * \; |hist| = front \\[10pt]
\textbf{match} \; \text{filter} \; (\lambda(front', \_) \cdot front' = front) \; prophs \; \textbf{with} \\[4pt]
\mid [] \Rightarrow \boxed{\bullet - \cdot \circ -}^{\gamma.\text{winner}} \\[4pt]
\mid (\_, id) :: \_ \Rightarrow \\[4pt]
\quad \bigvee \left[
\begin{array}{l}
\boxed{\bullet - \cdot \circ -}^{\gamma.\text{winner}} \\[6pt]
\text{identifier} \; id * \exists \, \Phi \cdot \boxed{\bullet \, (front, \Phi)}^{\gamma.\text{winner}} \quad * \, \text{chaselev-au} \; \gamma \; \iota \; \Phi
\end{array}
\right.
\end{array}
\right.
$$

# State

$\text{chaselev-state}_{3,1}\ \gamma\ \textit{front back hist prophs} \triangleq$

$$* \left[ \begin{array}{l} \textit{front} = \textit{back} \\ \boxed{\bullet\, \textit{hist}}^{\gamma.\text{hist}} * |\textit{hist}| = \textit{front} + 1 \\[2ex] \textbf{match filter } (\lambda(\textit{front}', \_) \cdot \textit{front}' = \textit{front})\ \textit{prophs}\ \textbf{with} \\ |\ [] \Rightarrow \boxed{\circ\,(\textit{front}, -)}^{\gamma.\text{winner}} \\ |\ \_ \Rightarrow \exists\, \Phi \cdot \boxed{\bullet\,(\textit{front}, \Phi)}^{\gamma.\text{winner}} * \Phi(\texttt{SOME}\ \textit{hist}[\textit{front}]) \end{array} \right]$$

# State

$$\text{chaselev-state}_{3,2} \; \gamma \; front \; back \; hist \triangleq$$

$$* \begin{bmatrix} front = back + 1 \\ \boxed{\bullet \; hist}^{\gamma.\text{hist}} \quad * \; |hist| = front \\ \boxed{\bullet \; - \cdot \circ \; -}^{\gamma.\text{winner}} \end{bmatrix}$$

# Prophecy variable with memory

TODO