



# Verification of Chase-Lev work-stealing deque

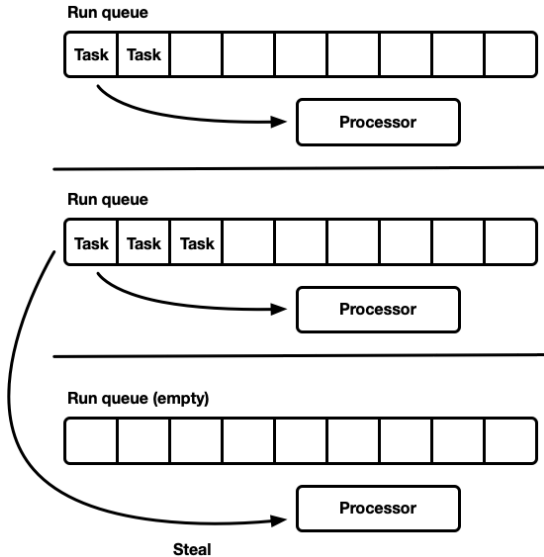
Clément Allain  
François Pottier

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

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



## 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

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

## 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

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

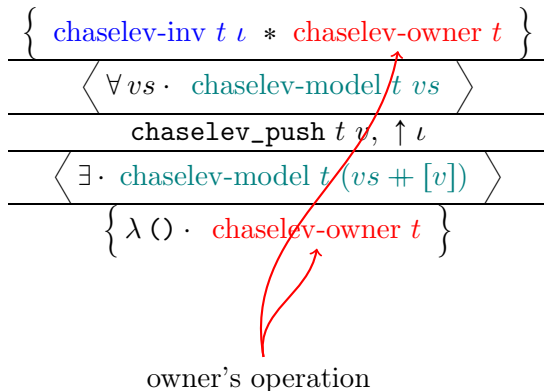
$Q_{\text{lin}}$  : public postcondition

## Specification — chaselev\_push

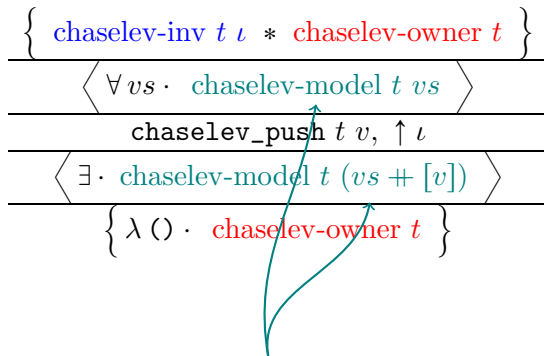
For a concurrent data structure:

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

## Specification — chaselev\_push



## Specification — chaselev\_push

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


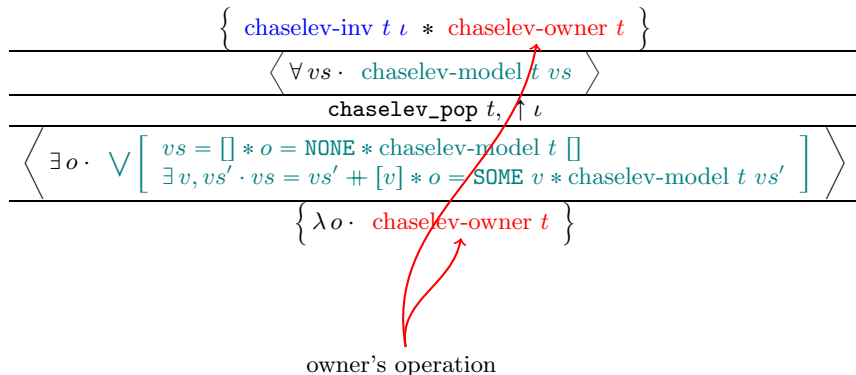
$v$  is atomically pushed at the owner's end



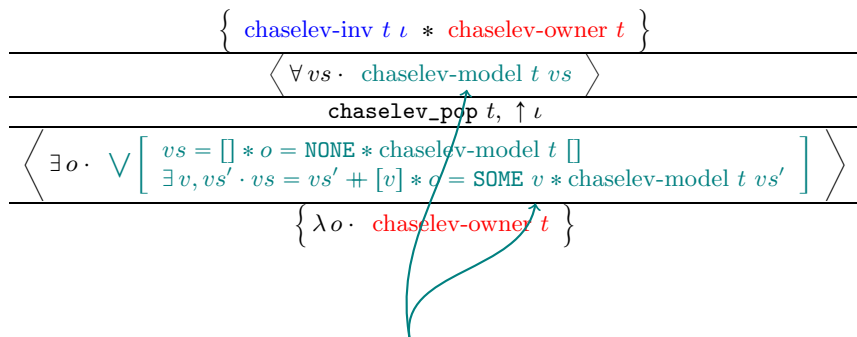
## Specification — chaselev\_pop

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

# Specification — chaselev\_pop



## Specification — chaselev\_pop

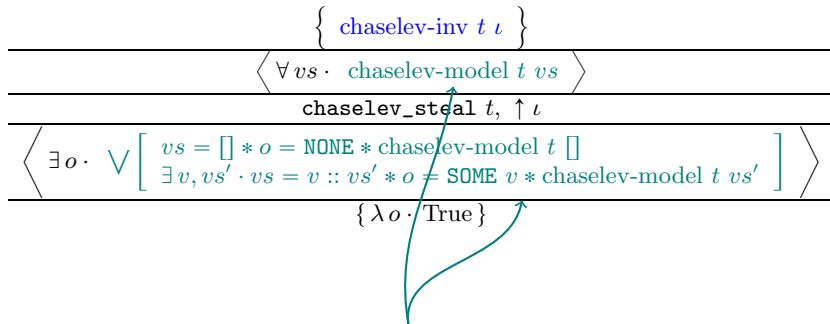


either 1) the deque is seen empty  
or 2) some value  $v$  is atomically popped at the owner's end

## 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 {

CHASELEVFRONTVALID

$$\frac{\boxed{\bullet \textit{front}_1}^{\gamma.\textit{front}} \quad \boxed{\circ \textit{front}_2}^{\gamma.\textit{front}}}{\textit{front}_2 \leq \textit{front}_1}$$

data: in

front: m

CHASELEVFRONTUPDATE

$$\frac{\textit{front} \leq \textit{front}' \quad \boxed{\bullet \textit{front}}^{\gamma.\textit{front}}}{\boxed{\bullet \textit{front}'}^{\gamma.\textit{front}}}$$

CHASELEVFRONTFRAGGET

$$\frac{\boxed{\bullet \textit{front}}^{\gamma.\textit{front}}}{\boxed{\circ \textit{front}}^{\gamma.\textit{front}}}$$



## Physical state

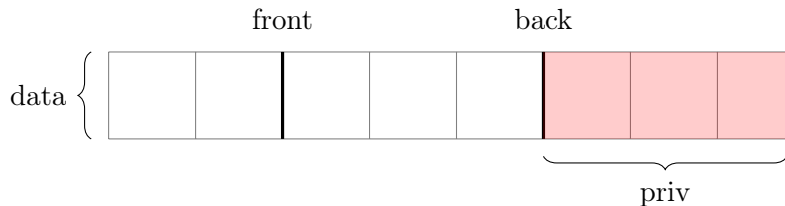


**data:** infinite array storing all values

**front:** *monotone* index for thieves' end

**back:** index for owner's end

# Physical state



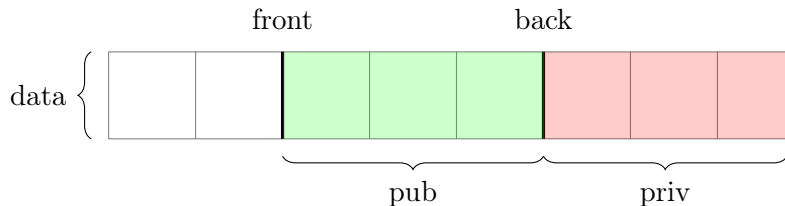
**data:** infinite array storing all values

**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

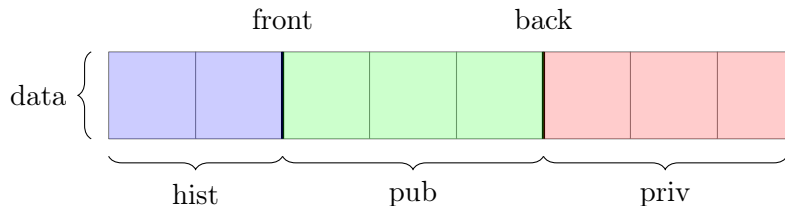
**front:** *monotone* index for thieves' end

**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

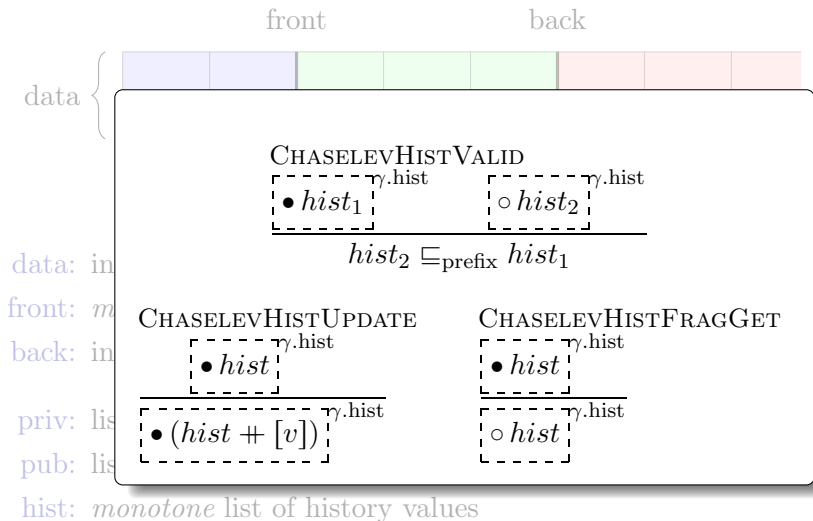
**back:** index for owner's end

**priv:** list of private values (controlled by owner)

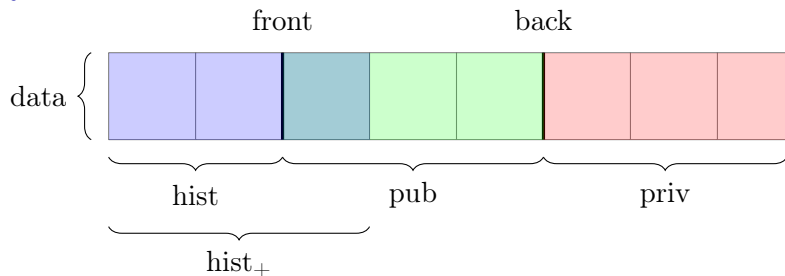
**pub:** list of public values (= model)

**hist:** *monotone* list of history values

# Physical state



## Physical state



**data:** infinite array storing all values

**front:** *monotone* index for thieves' end

**back:** index for owner's end

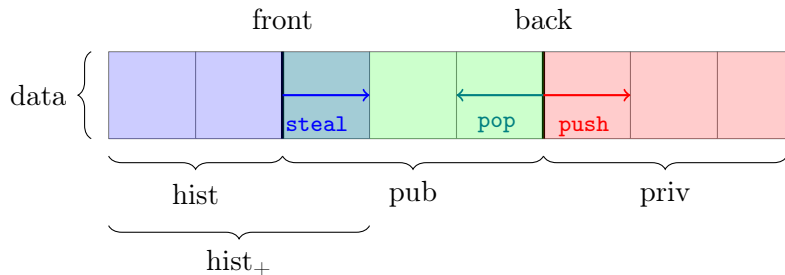
**priv:** list of private values (controlled by owner)

**pub:** list of public values (= model)

**hist:** *monotone* list of history values

**hist<sub>+</sub>:** *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)

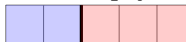
**pub:** list of public values (= model)

**hist:** *monotone* list of history values

**hist<sub>+</sub>:** *monotone* list of extended history values

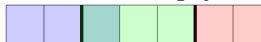
# Logical state

① empty



front = back

② non-empty

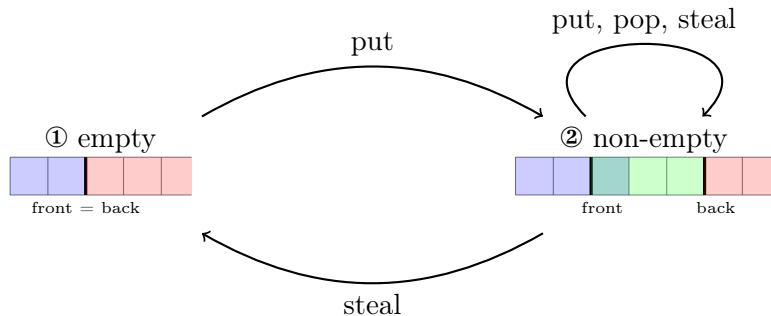


front

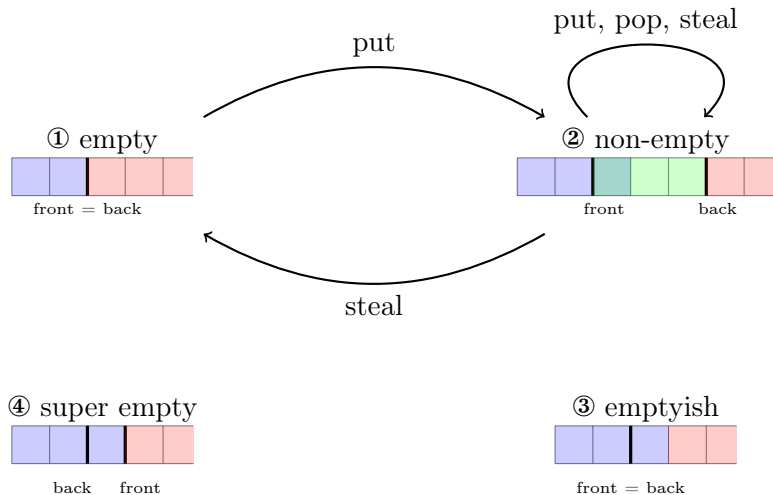
back



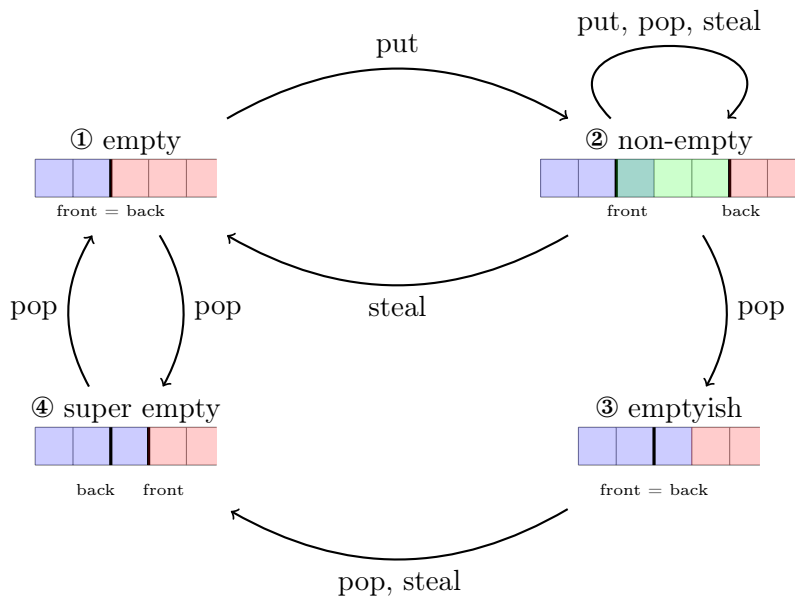
# Logical state



# 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

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

# Invariant

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

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

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

# State

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

# State

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

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

# State

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

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

# State

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

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

# State

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

$$* \left[ \begin{array}{l} \text{front} = \text{back} + 1 \\ \begin{array}{|l} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \end{array} \gamma.\text{hist} \\ \bullet \text{ hist} \end{array} \right] * |\text{hist}| = \text{front}$$

$$\left[ \begin{array}{|l} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \end{array} \gamma.\text{winner} \\ \bullet \text{ --- } \circ \text{ ---} \end{array} \right]$$

# Prophecy variable with memory

TODO