

Verification of Chase-Lev work-stealing deque

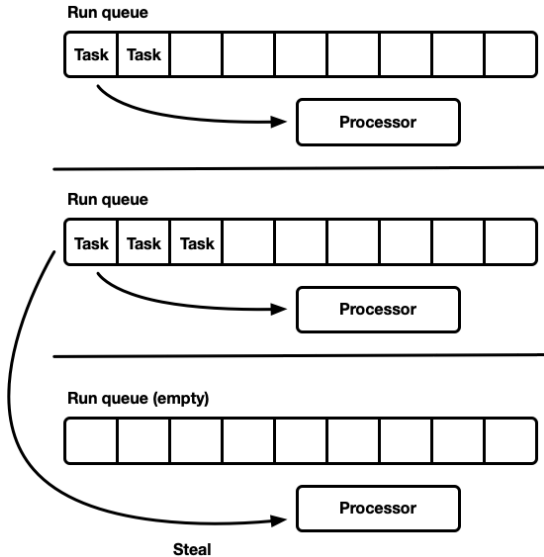
Clément Allain
François Pottier

April 24, 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

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

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

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

P : private precondition

Q : private postcondition

P_{lin} : public precondition

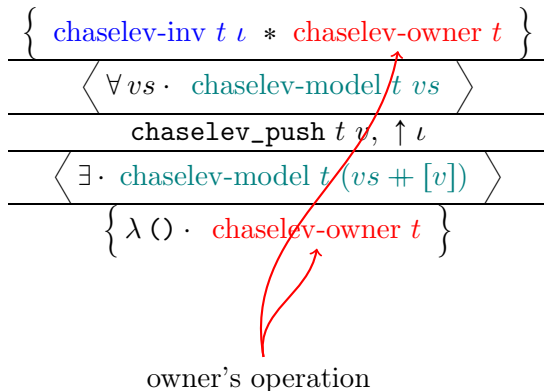
Q_{lin} : public postcondition

Specification — chaselev_push

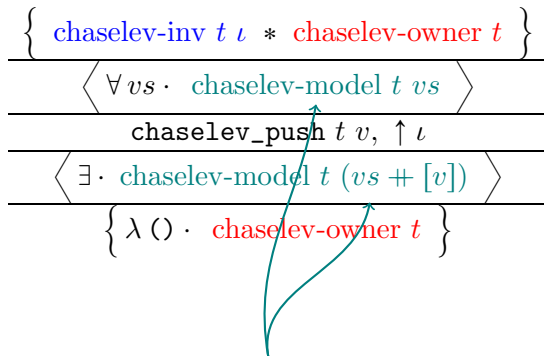
For a concurrent data structure:

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

Specification — chaselev_push



Specification — chaselev_push

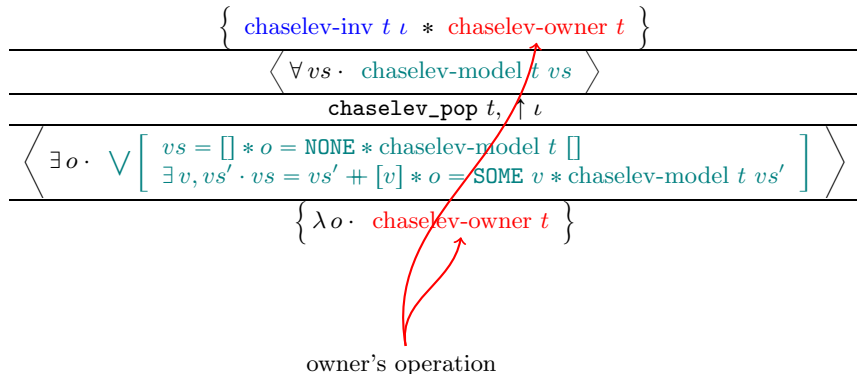


some value v is atomically pushed at the owner's end

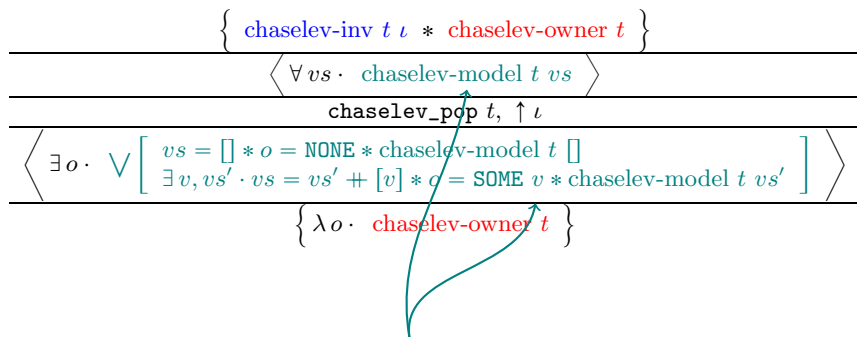
Specification — chaselev_pop

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

Specification — chaselev_pop



Specification — chaselev_pop

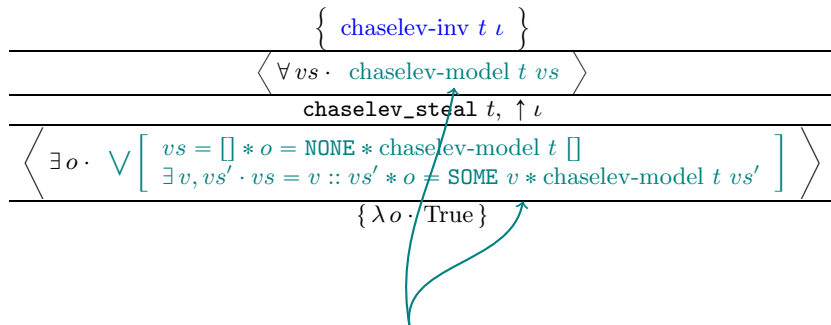


either 1) some value v is atomically popped at the owner's end
or 2) the deque is seen empty

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

Thank you for your attention!