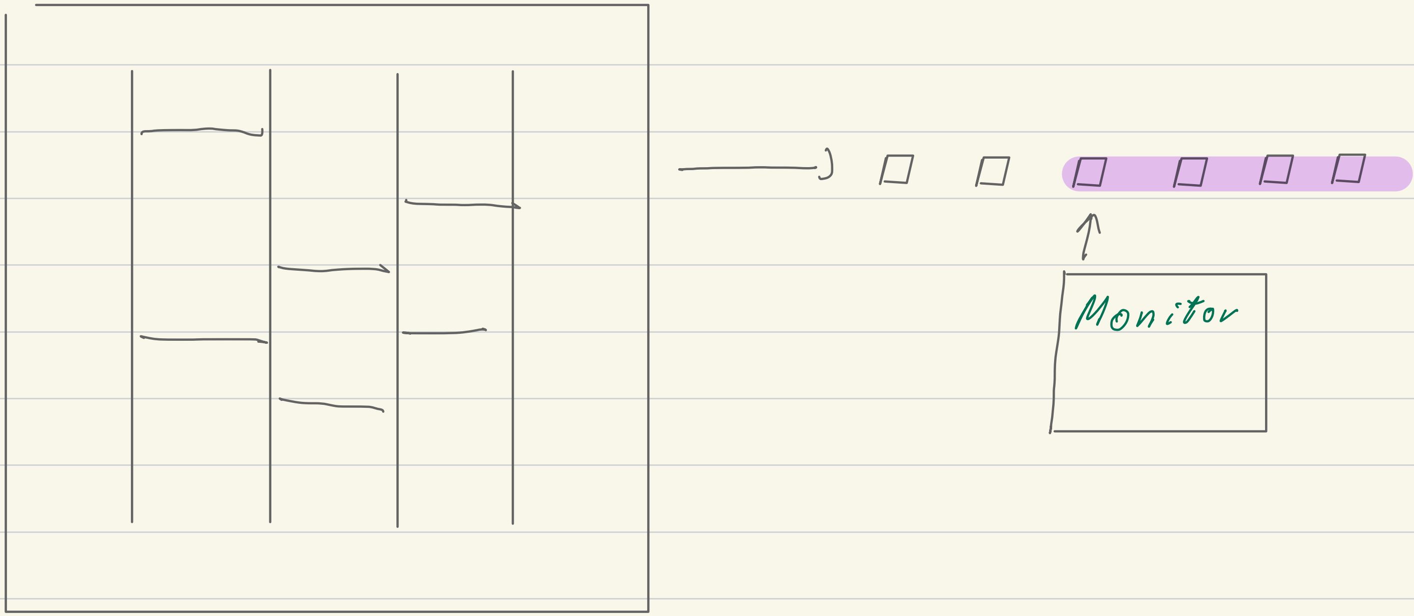
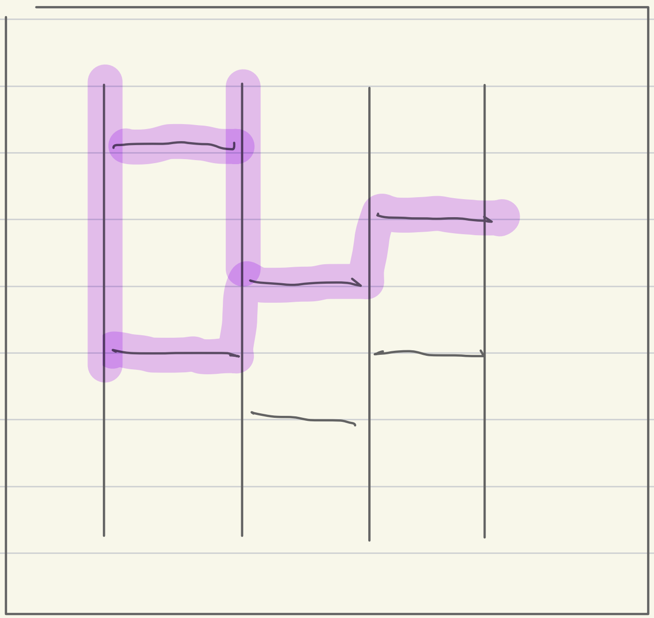
Distributed race detection

Igor Walukiewicz





Self monitoring program

Distributed monitor

Race prediction

- Operations $r_t(x)$, $w_t(x)$
  $acq_t(\ell)$, $rel_t(\ell)$

- Lock discipline :
  critical sections of the
  same lock are disjoint

	t_1	t_2
1	$\mathsf{acq}(\ell)$	
2	$w(y)$	
3	$w(x)$	
4		$r(x)$
5	$\mathsf{rel}(\ell)$	
6		$\mathsf{acq}(\ell)$
7		$w(y)$
8		$\mathsf{acq}(\ell)$

Race prediction

- Operations $r_t(x)$, $w_t(x)$
  $acq_t(\ell)$, $rel_t(\ell)$

- Lock discipline:
  critical sections of the
  same lock are disjoint

- Race $(w_t(x), r_{t'}(x))$ or $(w_t(x), w_{t'}(x))$
  the two events are concurrent in σ
  (can appear next to each other
  in some sound reordering of σ)

	t_1	t_2
1	acq(ℓ)	
2	$w(y)$	
3	$w(x)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

Race prediction

- Operations $r_t(x)$, $w_t(x)$
  $acq_t(\ell)$, $rel_t(\ell)$
- Lock discipline:
  critical sections of the
  same lock are disjoint
- Race $(w_t(x), r_{t'}(x))$ or $(w_t(x), w_{t'}(x))$
  the two events are concurrent in σ
  (can appear next to each other
  in some sound reordering of σ)

Race prediction problem:
looking at σ detect if
there is a race

	t_1	t_2
1	acq(ℓ)	
2	$w(y)$	
3	$w(x)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

Race prediction

- Operations $r_t(x)$, $w_t(x)$
  $acq_t(\ell)$, $rel_t(\ell)$

- Lock discipline:
  critical sections of the
  same lock are disjoint

- Race $(w_t(x), r_{t'}(x))$ or $(w_t(x), w_{t'}(x))$
  the two events are concurrent in σ
  (can appear next to each other
  in some sound reordering of σ)

Race prediction problem:
looking at σ detect if
there is a race

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3		$w(z)$
4	rel(ℓ)	
5		$w(x)$

Execution σ
No data race

	t_1	t_2
1	acq(ℓ)	
2		$w(z)$
3	$r(x)$	
4		$w(x)$
5	rel(ℓ)	

Execution σ'

Happens-before order [Lamport]

What is a sound reordering of σ?

- operations on different threads permute unless they concern the same lock

$e_1 \leq^{\sigma}_{HB} e_2$ is the smallest partial-order s.t.

- events of the same thread are ordered
- if $\ldots \mathrm{rel}_t(\ell) \ldots \mathrm{acq}_{t'}(\ell) \ldots$ then $\leq^{\sigma}_{HB}$

	t_1	t_2	t_3
1	$\mathrm{acq}(\ell_1)$		
2	$w(x)$		
3	$\mathrm{rel}(\ell_1) \xrightarrow{HB}$		
4		$\mathrm{acq}(\ell_1)$	
5		$\mathrm{acq}(\ell_2)$	
6		$\mathrm{rel}(\ell_2) \xrightarrow{HB}$	
7			$\mathrm{acq}(\ell_2)$
8			$w(x)$
9		$\mathrm{rel}(\ell_1)$	
10			$\mathrm{rel}(\ell_2)$

Happens-before order [Lamport]

What is a sound reordering of σ?

- operations on different threads permute unless they concern the same lock

$e_1 \leq^{\sigma}_{HB} e_2$ is the smallest partial-order s.t.

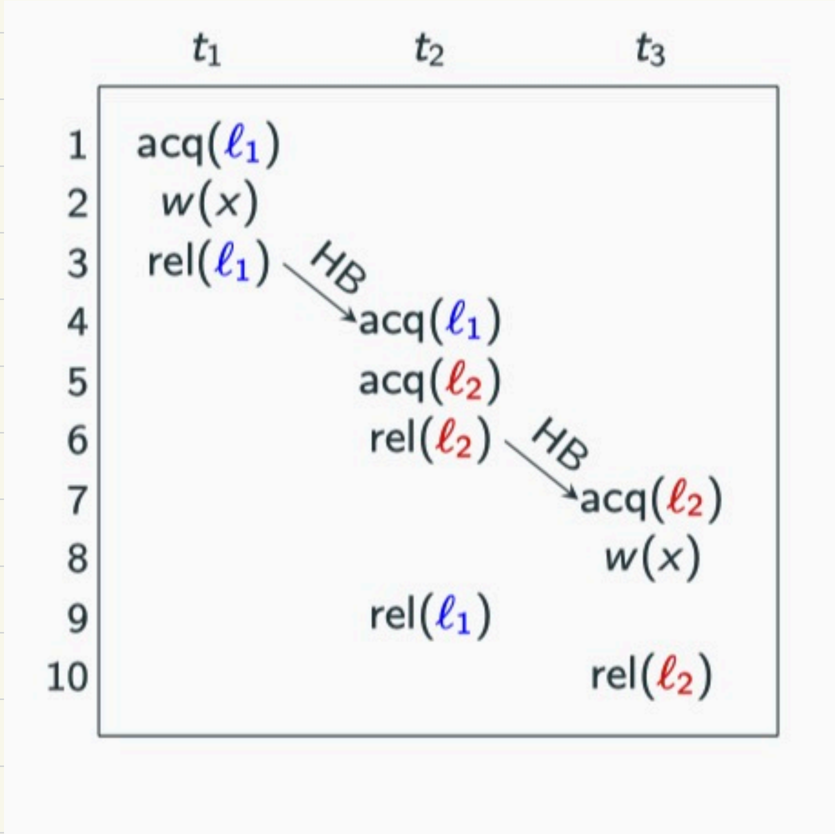
- events of the same thread are ordered
- if $\ldots \, rel_t(\ell) \ldots acq_{t'}(\ell) \ldots$ then $\leq^{\sigma}_{HB}$

$\sigma \approx_{HB} \sigma'$ if the two have the same set of events and give the same HB order

Race detection : race prediction for $\approx_{HB}$

By looking at σ detect if there is $\sigma' \approx_{HB} \sigma$ with a race situation

$w_t(x) \; r_{t'}(x)$



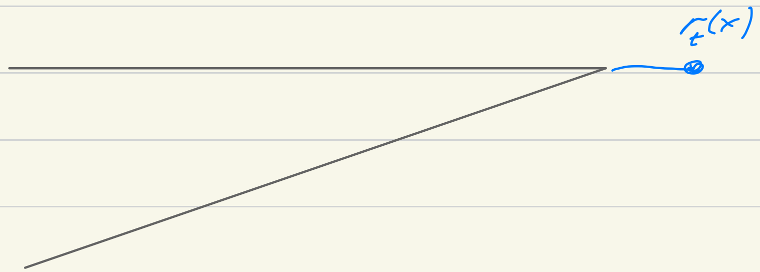
Vector clocks [Lamport]

Clocks measuring HB-time

$\mathcal{C}_t : \mathcal{T} \to \mathbb{N}$ $\quad$ $\mathcal{C}_t(t)$ - program counter of t

$\mathcal{C}_t(t')$ - the last program counter of t' seen by t

$r_t(x)$, $w_t(x)$



$\mathcal{C}_t(t) := \mathcal{C}_t(t) + 1$

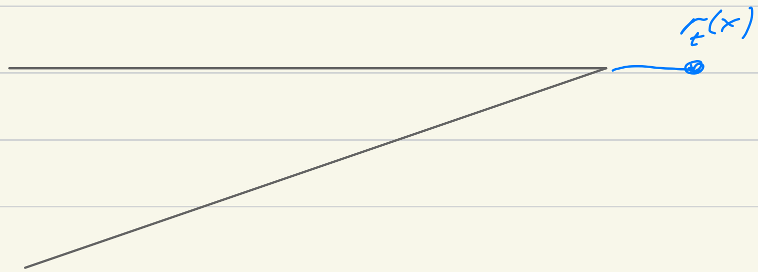
Vector clocks [Lamport]

Clocks measuring HB-time

$\mathcal{C}_t : \mathcal{T} \to \mathbb{N}$ $\quad$ $\mathcal{C}_t(t)$ - program counter of t

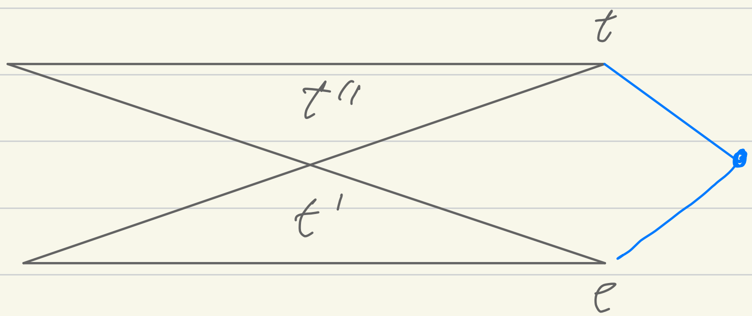
$\mathcal{C}_t(t')$ - the last program counter of t' seen by t

$r_t(x)$, $w_t(x)$



$\mathcal{C}_t(t) := \mathcal{C}_t(t) + 1$

$acq_t(\ell)$



$\mathcal{C}_t(t) := \mathcal{C}_t(t) + 1$

$acq_t(\ell)$

$\forall t' \neq t \quad \mathcal{C}_t(t') = \max(\mathcal{C}_t(t'), \mathcal{C}_\ell(t'))$

$\forall t' \quad \mathcal{C}_\ell(t') =$ —— " ——

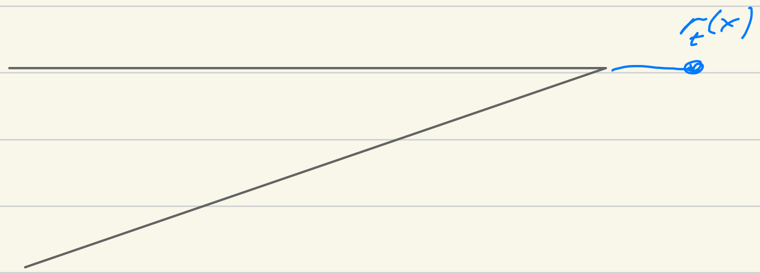
Vector clocks [Lamport]

Clocks measuring HB-time

$C_t : T \to \mathbb{N}$ $\quad C_t(t)$ - program counter of t

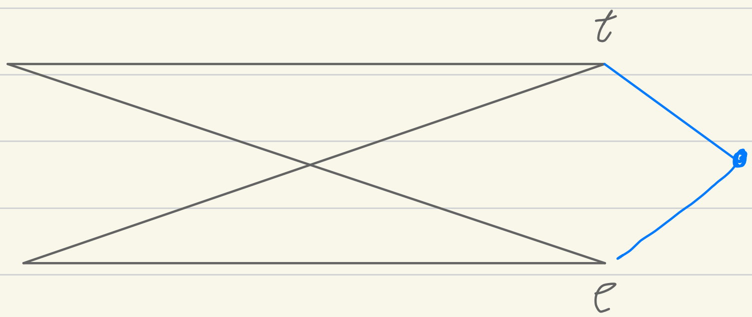
$C_t(t')$ - the last program counter of t' seen by t

$r_t(x)$, $w_t(x)$



$C_t(t) := C_t(t)+1$

$acq_t(e)$



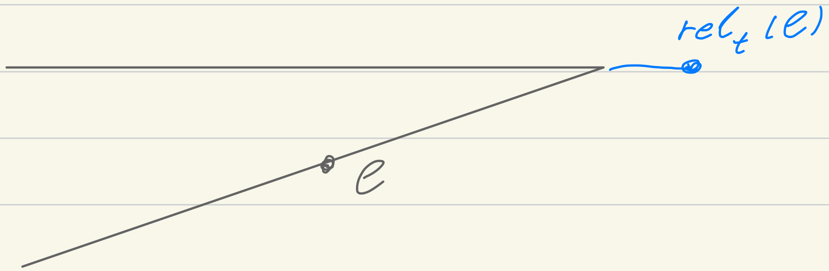
$C_t(t) := C_t(t)+1$

$acq_t(e)$

$\forall t' \neq t \quad C_t(t') = \max(C_t(t'), C_e(t'))$

$\forall t' \quad C_e(t') =$ —— " ——

$rel_t(e)$



$C_t(t) := C_t(t)+1$

$\forall t' : C_e(t') = C_t(t')$

Race detection

$\mathcal{C}_t : \mathcal{T} \to \mathbb{N}$ $\mathcal{C}_t(t)$ - program counter of t

$\mathcal{C}_t(t')$ - the last program counter of t' seen by t

$W^x \in \mathcal{T} \times \mathbb{N}$ thread that made the last write to x, and its PC

$R^x : \mathcal{T} \to \mathbb{N}$ for every thread last read from x

Race check

$r_t(x)$ if $W^x = (t', i)$ and $\mathcal{C}_t(t') < i$ then race else $R^x(t) := \mathcal{C}_t(t)$

$w_t(x)$ if ——— // ———

or $\exists t'.\ \mathcal{C}_t(t') < R^x(t')$ then race else $W^x = (t, \mathcal{C}(t))$

Race detection

Lamport. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. Commun. ACM 21

Itzkovitz, Schuster, Zeev-Ben-Mordehai. 1999. Toward Integration of Data Race Detection in DSM Systems. J. Parallel Distrib. Comput.

Flanagan, Freund. 2009. FastTrack: Efficient and Precise Dynamic Race Detection. PLDI ’09

Thokair, Zhang, Mathur, Viswanathan, Dynamic Race Detection with O(1) Samples, POPL'23

Race detection (distributed)

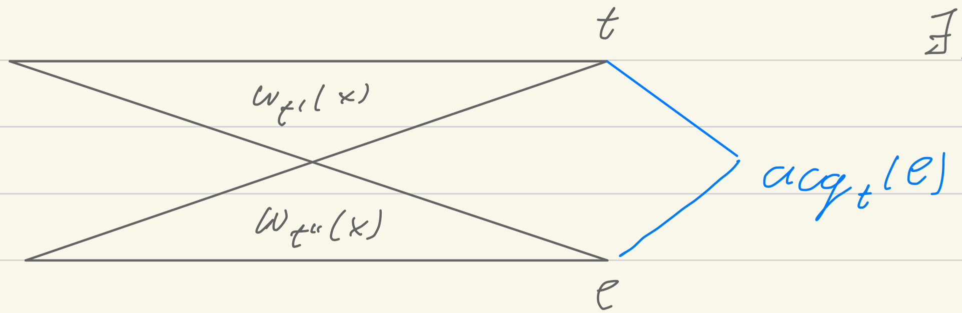
$C_t : T \to \mathbb{N}$ $\quad$ $C_t(t)$ - program counter of t

$C_t(t')$ - the last program counter of t' seen by t

$W_t^x \in T \times \mathbb{N}$ $\quad$ thread that made the last write to x, and its PC

$R_t^x : T \to \mathbb{N}$ $\quad$ for every thread last read from x

Race check



$\exists x \quad W_t(x) = (t', i) \quad C_e(t') < i$

$W_e(x) = (t'', j) \quad C_t(t'') < j$

Race detection (distributed)

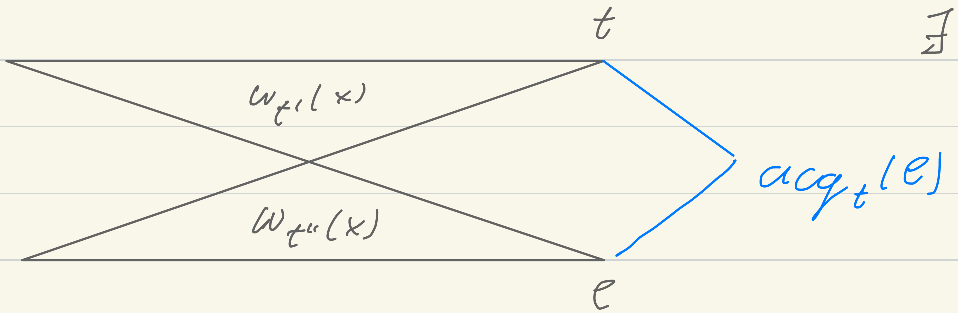
$\tau_t : T \to \mathbb{N}$ $\quad$ $\tau_t(t)$ - program counter of t

$\tau_t(t')$ - the last program counter of t' seen by t

$W_t^x \in T \times \mathbb{N}$ $\quad$ thread that made the last write to x, and its PC

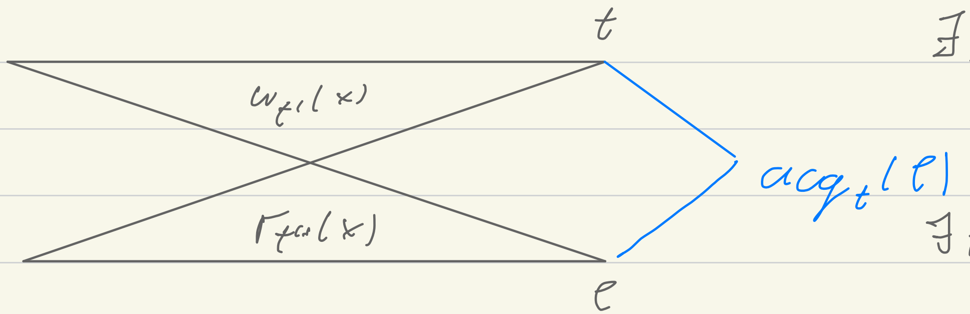
$R_t^x : T \to \mathbb{N}$ $\quad$ for every thread last read from x

Race check



$\exists x \quad W_t(x) = (t', i) \quad \tau_e(t') < i$

$W_e(x) = (t'', j) \quad \tau_t(t'') < j$



$\exists x \quad W_t(x) = (t', i) \quad \tau_e(t') < i$

$\exists t'' \quad R(x, t'') > \tau_e(t')$

Race detection (distributed)

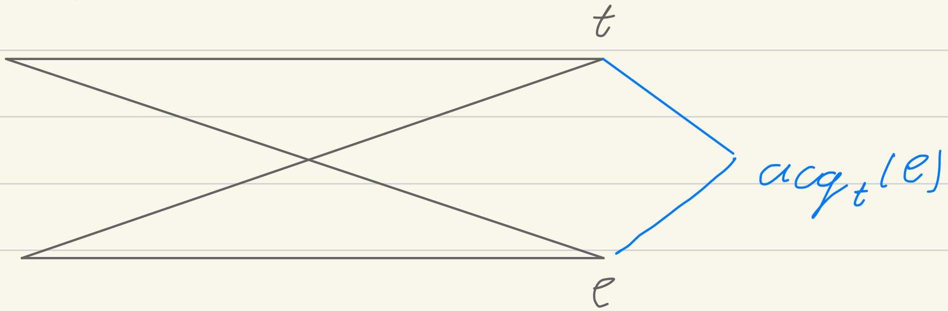
$C_t : T \to \mathbb{N}$ $\quad$ $C_t(t)$ - program counter of t

$C_t(t')$ - the last program counter of t' seen by t

$W_t^x \in T \times \mathbb{N}$ $\quad$ thread that made the last write to x, and its PC

$R_t^x : T \to \mathbb{N} \cup \{\bot\}$ for every thread last read from x

Update is difficult



for $x \in Var$

$\quad W_t^x := (t', i)$

$\quad$ if $C_e(t') < i$ then $W_e^x := (t', i)$

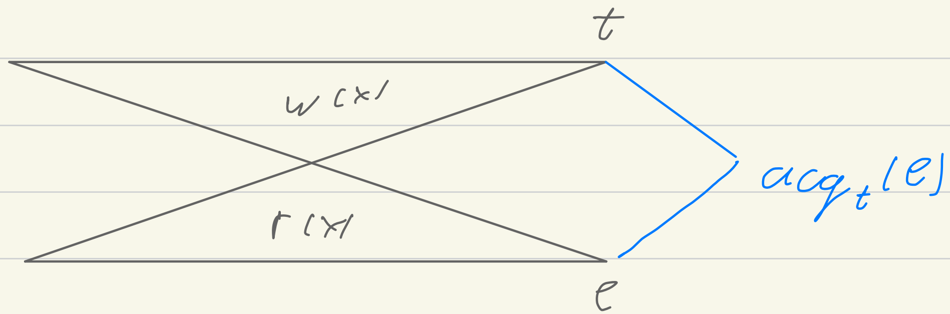
for $t' \in T$

$\quad$ if $R_t^x(t') > C_e(t')$ then $R_e^x(t') = R_t^x(t')$

Distributed race detection, take 2

$W_t : \mathcal{T} \to (\mathbb{N} \times Vars)^*$

$R_t : \mathcal{T} \to (\mathbb{N} \times Vars \times Vals)^*$



Race if there is $x \in Vars$ $t_1, t_2 \in \mathcal{T}$ such that

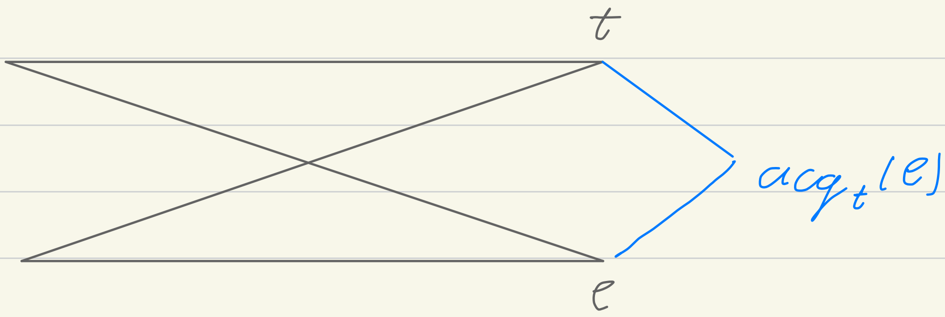
- $(i, x) \in W_t(t_1)$
- $e_e(t_1) < i$
- $(j, x, \alpha) \in R_e(t_2)$
- $e_t(t_2) < j$

Distributed race detection, take 2

$W_t : J \to (\mathbb{N} \times Vars)^*$

$R_t : J \to (\mathbb{N} \times Vars \times Vals)^*$

$W_t(t')$ sorted by $\mathbb{N}$



Race if there is $x \in Vars$ $t_1, t_2 \in J$ such that

- $(i, x) \in W_t(t_1)$
- $C_e(t_1) < i$
- $(j, x, \alpha) \in R_e(t_2)$
- $C_t(t_2) < j$

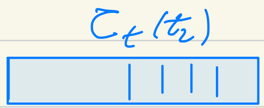
t_1 moved in $t{\downarrow} - e{\downarrow}$ if $C_t(t_1) > \tau_e(t_1)$ $\mapsto$ $t_1 \in MI_t$

x written in $t{\downarrow} - e{\downarrow}$ if $(i, x) \in W_t(t_1)$ $i > \tau_e(t_1)$ $\mapsto$ Wx_t

for $t_2 \in MI_e$

  for $(i, x, \alpha) \in R_e(t_2)$ with $i > \tau_t(t_2)$

    if $x \in Wx_t$ then race

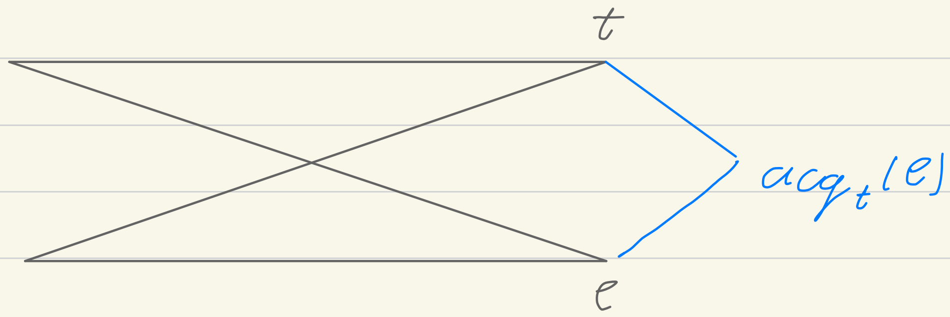


$R_e(t_2)$

Distributed race detection, take 2

$W_t : \mathcal{T} \to (\mathbb{N} \times Vars)^*$

$R_t : \mathcal{T} \to (\mathbb{N} \times Vars \times Vals)^*$



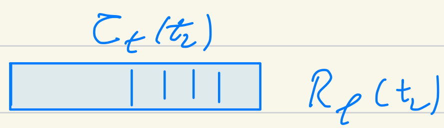
Race if there is $x \in Vars$ $t_1, t_2 \in \mathcal{T}$ such that

- $(i, x) \in W_t(t_1)$
- $\tau_e(t_1) < i$
- $(j, x, \alpha) \in R_e(t_2)$
- $\tau_t(t_2) < j$

t_1 moved in $t\downarrow - e\downarrow$ if $\tau_t(t_1) > \tau_e(t_1)$ $\mapsto$ $t_1 \in MI_t$

x written in $t\downarrow - e\downarrow$ if $(i, x) \in W_t(t_1)$ $i > \tau_e(t_1)$ $\mapsto$ Wx_t

for $t_2 \in MI_e$
    for $(i, x, \alpha) \in R_e(t_2)$ with $i > \tau_t(t_2)$
        if $x \in Wx_t$ then race



The work is proportional to the size of symmetric difference.

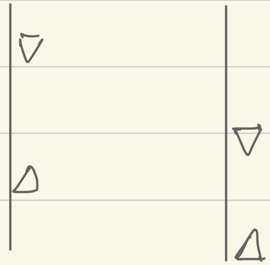
TODO

- Faster distributed monitoring
- Distributed monitoring for other properties
    - serializability

    - any Σ_2 property

TODO

- Faster distributed monitoring
- Distributed monitoring for other properties
  - serializability



  - any Σ_2 property
- Race prediction
  reads from equivalence