Vector clock

A *vector clock* is a tuple of N logical clocks, one per thread:

$$(c_0, c_1, \ldots, c_{N-1})$$

Equivalently: vector clock is a mapping from *Threads* to natural numbers

If V is a vector clock, $V(t)$ denotes logical clock of thread t

Set of all vector clocks: VC

Vector clock

Bottom vector clock: $\bot = (0, 0, \ldots, 0)$

Every thread has logical clock 0

Partial order on vector clocks: $V_1 \sqsubseteq V_2$ if and only if $\forall t \,.\, V_1(t) \leq V_2(t)$

One vector clock is "less-than or equal to" another vector clock if logical clocks are pointwise less-than or equal

Join of vector clocks: $V_1 \sqcup V_2$ = pointwise maximum of V_1, V_2

Increment: $inc_t(V) = V[t \mapsto V(t) + 1]$

Same as V, but logical clock for t is incremented

Purpose of per-thread vector clocks

$C : Threads \rightarrow VC$	Each *thread* has a vector clock

C_t - shorthand for $C(t)$

C_t represents what thread t knows about the logical clocks of all threads

If I am thread t, then:

- $C_t(t)$ is *my* logical clock. It will always be **positive**
- For $u \neq t$, $C_t(u) = z$ means "I know that thread u's logical clock is at least z"

When t **acquires a lock**, t gets information about the logical clocks of threads who previously held the lock

Purpose of per-lock vector clocks

$L : Locks \rightarrow VC$	Each *lock* has a vector clock

L_m - shorthand for $L(m)$

L_m represents the logical clock each thread had last time it **released** m

For a lock m and thread t:

Remember that a thread's true logical clock value will always be positive

- $L_m(t) = 0$ means that t has never released m
- Otherwise $L_m(t)$ was t's logical clock last time t released m

Purpose of per-location read vector clocks

$R : Locations \rightarrow VC$	Each *location* has a *read* vector clock

R_x - shorthand for $R(x)$

R_x represents the logical clock each thread had last time it **read from** x

For a location x and thread t:

- $R_x(t) = 0$ means that t has never read from x
- Otherwise $R_x(t)$ was t's logical clock last time t read from x

Purpose of per-location write vector clocks

$W : Locations \rightarrow VC$ Each *location* has a *write* vector clock

W_x - shorthand for $W(x)$

W_x represents the logical clock each thread had last time it **wrote to** x

For a location x and thread t:

- $W_x(t) = 0$ means that t has never written to x
- Otherwise $W_x(t)$ was t's logical clock last time t wrote to x

Initial analysis state

- $C = (inc_0(\bot), inc_1(\bot), \ldots, inc_{N-1}(\bot))$

  Thread t knows its logical clock is 1, and thinks all other threads' logical clocks are 0

- $L = \lambda\, m\, .\, \bot$

  No thread has unlocked any mutex

- $R = \lambda\, x\, .\, \bot$
- $W = \lambda\, x\, .\, \bot$

  No thread has read from or written to any location

Vector clock algorithm state

Recall:

- N threads, starting from 0
- *Locks*: available locks for synchronisation
- *Locations*: possibly-shared locations, on which races could occur

Algorithm state: (C, L, R, W):

- $C : Threads \rightarrow VC$ — Each *thread* has a vector clock
- $L : Locks \rightarrow VC$ — Each *lock* has a vector clock
- $R : Locations \rightarrow VC$ — Each *location* has a *read* vector clock
- $W : Locations \rightarrow VC$ — Each *location* has a *write* vector clock