We consider a contention-aware counter consisting of K registers $\{a_i\}_{i=1}^{i=K}$. $inc(delta)$ operation choses a random register and atomically increments it by a non-negative $delta$. get operation summarizes values of the registers in order from a_1 to a_K.

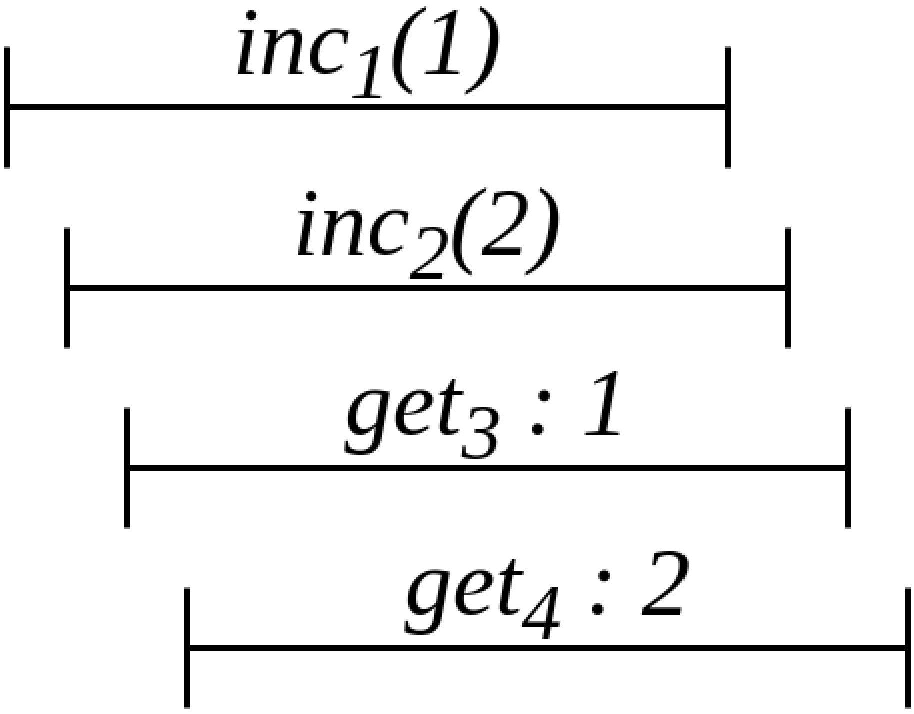
We use the following notion for histories: $H = \{ev_i\}$, $ev_i = \begin{cases} start.\, op_i \\ end.\, op_i \end{cases}$,
$op_i = \begin{cases} get_i : result \\ inc_i(delta) \end{cases}$.

Statement. The counter is not linearizable.

Proof. Consider an example with $K = 2$ and the following concurrent history. Lines written with regular text are atomic steps made by the operations, they are not part of the history.

$\mathbf{start.\, inc_1(1)}$
$\mathbf{start.\, inc_2(2)}$
$\mathbf{start.\, get_3}$
$\mathbf{start.\, get_4}$
$get_3\ reads\ a_1 \to 0$
$inc_2\ increments\ a_1\ by\ 2$
$get_4\ reads\ a_1 \to 2$
$get_4\ reads\ a_2 \to 0$
$inc_1\ increments\ a_2\ by\ 1$
$get_3\ reads\ a_2 \to 1$
$\mathbf{end.\, inc_1}$
$\mathbf{end.\, inc_2}$
$\mathbf{end.\, get_3 : 1}$
$\mathbf{end.\, get_4 : 2}$

This history consists of four concurrent operations: $inc_1(1), inc_2(2), get_3 : 1, get_4 : 2$. Imagine these inc operations being performed sequentially. Then expected values of the counter would be either $0 \to 1 \to 3$, or $0 \to 2 \to 3$. So the history in which gets yield 1 and 2 cannot be linearized.



Moreover, this history is not sequentially consistent.

Let $<_H$ be a an order induced by the history H on operations:

$op_1 <_H op_2$ iff $end.op_1$ precedes $start.op_2$ in H.

Let I be a set of all inc operations. For operation $get_i : result$ let $L_i = \{inc_j \in I | inc_j <_H get_i\}$, $J_i = \{inc_j \in I | get_i <_H inc_j\}$, $C_i = \{inc_j \in I | get_i ||_H inc_j\} = I \setminus (L_i \cup J_i)$.

For a set $U \subset I$ let $sum(U) = \Sigma_{inc_i:delta_i \in U} delta_i$. For example, $sum(L_i)$ is the sum of all increments completed before the start of get_i.

Statement. For any concurrent history H and a $get_i : res_i$ operation in it $sum(L_i) \leq res_i \leq sum(L_i) + sum(C_i)$.

Proof. We rephrase lemma 1.1 and lemma 1.2 from PROOF.pdf.

Lemma 1.1. For any operation $get_i : result\ result \geq sum(L_i)$. **Proof.** Fetch-and-adds performed by operations in L_i happens-before reads performed by get_i. As registers are never decremented, the reads return at least those values which are written by the last FAA to the corresponding register performed by an operation from L_i. Sum of these values is equal to $sum(L_i)$.

Lemma 1.2. For any operation $get_i : result$
$result \leq sum(I) - sum(J_i) = sum(L_i) + sum(C_i)$. **Proof.** Reads performed by get_i happens-before fetch-and-adds performed by operations in J_i. As registers are never decremented, the reads return at most those values which are read by the first FAA to the corresponding register performed by an operation from J_i. Sum of these values is equal to $sum(I \setminus J_i)$.