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

Statement. The given counter is linearizable.

Proof. Consider a concurrent history $H = \{ev_i\}$, $ev_i = \begin{cases} start.\,op_i \\ end.\,op_i \end{cases}$, $op_i = \begin{cases} get_i : result \\ inc_i \end{cases}$.

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.

For sequential histories we will consider notions $\{op_1, op_2, \dots op_n\}$ and $\{start.\,op_1, end.\,op_1, start.\,op_2, end.\,op_2, \dots start.\,op_n, end.\,op_n\}$ to be equivalent.

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

Lemma 1.1. For any operation $get_i : result$ $result \geq |L_i|$. **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 the size of L_i.

Lemma 1.2. For any operation $get_i : result$ $result \leq |I| - |J_i| = |L_i| + |C_i|$, i.e. $result$ does not exceed the number of incs which completed before get_i or concurrent with it. **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 the size of $I \setminus J_i$.

Denote the number of inc operations in a set or sequence Q $incs(Q)$.

Next, we provide a procedure $linearize(H)$ to construct a sequential history H', given H.

```
linearize(H):
  input H - sequence of events
  let S = empty set of operations
  let H' = empty sequence of operations
  foreach event e in H:
    if e = start.op_i:
      add op_i to S
    else if e = end.op_i and op_i in S:
      if op_i = inc_i:
        foreach op_j in S: // block 1
          if op_j = get_j : res_j and res_j = incs(H'):
            remove get_j from S
            append get_j to H'
```

```
      else if op_i = get_i : res_i:
        while res_i < incs(H'): // block 2
          foreach op_j in S: // block 2.1
            if op_j = get_j : res_j and res_j = incs(H'):
              remove get_j from S
              append get_j to H'
          find inc_j in S // block 2.2
          remove inc_j from S
          append inc_j to H' // point a
      remove op_i from S
      append op_i to H'
  return H'
```

$linearize$ analyzes events from H in order and maintains a set of started but not yet finished operations S. When it encounters end of an operation op in H which is not yet contained in H', it appends some operations from S which must appear before op to H' and then appends op itself. Whenever an operation is appended to H' it is removed from S.

We now consider which operations from S are appended before the operation op.

- *(block 1)* If op is an inc operation, then all $get : result$ with $result = incs(H')$ must be appended to H' before this inc operation. So we take such gets from S and append them to H'.
- *(block 2)* If op is a $get : result$ operation, then exactly $result$ inc operations must appear before op in H'. So we take incs from S and append them to H' until $incs(H') = result$ *(block 2.2)*. Also, before that we take etgs with $result = incs(H')$ from S and append them to H' *(block 2.1)*.

Lemma 2. When an operation $get_i : res_i$ is added to S, $res_i \geq incs(H')$.

Proof. The operation is added to S when its start is encountered in H. To this moment all operations from L_i are appended to H' and none of operations from J_i are encountered and none of them are appended to H'.

Imagine that $incs(H') > res_i$. Consider an operation inc_j such that $incs(H')$ was equal to res_i before inc_j was appended to H'. According to lemma 1.1 $inc_j \in C_i$, i.e. it is concurrent with get_i. Then $end.\,inc_j$ is not yet encountered in H. Consequently, inc_j could be appended to H' only at *point a*. This means that an event $end.\,get_k : res_k$ appeared in H before $start.\,get_i : res_i$ for some operation get_k with $res_k > res_i$.

So there are two get operations $get_i : res_i$ and $get_k : res_k$, such that

- $get_k <_H get_i$,
- $res_k > res_i$.

All reads in get_k happens-before reads in get_i and values of the registers are never decremented. Consequently, all reads performed by get_i yield values greater than or equal to ones returned by reads in get_k, and $res_i \geq res_k$.

We came to a contradiction. Thus, $incs(H') \leq res_i$.

Lemma 3. Before execution of *block 2* $incs(H') \leq res_i$ and $incs(H') + incs(S) \geq res_i$.

Proof. Before execution of *block 2* get_i is contained in S. When it was added $incs(H')$ was less than or equal to res_i (lemma 3). Whenever an inc operation is appended to H', all get operations with result equal to $incs(H')$ are removed from S. Consequently, $incs(H') \leq res_i$. All operations from L_i are appended to H' before the end of get_i. Consider an operation $inc_j \in C_i$. It is either already appended to H' or contained in S. So $incs(H') + incs(S) = |L_i| + |C_i| \geq res_i$ (lemma. 1.2).

Corollary 1. After execution of *block 2* $incs(H') = res_i$.

Corollary 2. $linearize(H)$ is a legal sequential history. According to lemma 2 and corollary 1 all gets are appended to H' when number of inc operations in H' is equal to the result of the get operation.

Lemma 4. $linearize(H)$ is equivalent to H.

Proof. To prove equivalence of H and H' we have to prove, that total order $<_{H'}$ extends $<_H$. Consider two operations op_a and op_b, $op_a <_H op_b$. This means that $linearize$ encounters $end.op_a$ before $start.op_b$ and appends op_a to H' before it appends op_b to S. Consequently, $op_a <_{H'} op_b$.

Conclusion. According to corollary 2 and lemma 4 $linearize(H)$ is a linearization of H.