

Deque-Side Semantics

Disclaimer: we assume familiarity with [MLF+23], adopt its notational conventions, and steal its definitions! Further, we work with a specific subset of *Rio*, denoted **Rio**, namely

$$\frac{c \in \mathbf{Class}}{\mathbf{edf}[c], \mathbf{fifo}[c] \in \mathbf{Rio}} \text{ set2stream} \quad \frac{n \in \mathbb{N} \quad rs \in \mathbf{Rio}^n}{\mathbf{strict}[rs], \mathbf{rr}[rs] \in \mathbf{Rio}} \text{ stream2stream}$$

where **Class** is an opaque collection of *classes*.

1 Structure and Semantics of Rio Trees



Figure 1. Rio trees decorated by classes *A*, *B*, and *C*

Definition 1.1. For topology $t \in \mathbf{Topo}$, the set $\mathbf{RioTree}(t)$ of *Rio trees* over t is defined by

$$\frac{p \in \mathbf{PIFO}(\mathbf{Pkt}) \quad c \in \mathbf{Class}}{\text{Leaf}(p, c) \in \mathbf{RioTree}(*)} \quad \frac{ts \in \mathbf{Topo}^n \quad \forall 1 \leq i \leq n. qs[i] \in \mathbf{RioTree}(ts[i])}{\text{Internal}(qs) \in \mathbf{RioTree}(\text{Node}(ts))}$$

These are trees with leaves decorated by both classes and PIFOs.

Definition 1.2. For topology $t \in \mathbf{Topo}$, the set $\mathbf{OrdTree}(t)$ of *ordered trees* over t is defined by

$$\frac{}{\text{Leaf} \in \mathbf{OrdTree}(*)} \quad \frac{ts \in \mathbf{Topo}^n \quad rs \in \mathbf{Rk}^n \quad \forall 1 \leq i < j \leq n. rs[i] \neq rs[j] \quad \forall 1 \leq i \leq n. os[i] \in \mathbf{OrdTree}(ts[i])}{\text{Internal}(rs, os) \in \mathbf{OrdTree}(\text{Node}(ts))}$$

These are trees with each internal node's child given a *unique* rank, thereby inducing a total ordering of children.

Let $\mathbf{flow} : \mathbf{Pkt} \rightarrow \mathbf{Class}$ be an opaque mapping from packets to the class they belong to (flow inference).

Definition 1.3. For $t \in \mathbf{Topo}$, define $\mathbf{push} : \mathbf{RioTree}(t) \times \mathbf{Pkt} \times \mathbf{Rk} \rightarrow \mathbf{RioTree}(t)$ such that

$$\frac{\mathbf{flow}(\text{pkt}) = c \quad \text{push}(p, \text{pkt}, r) = p'}{\text{push}(\text{Leaf}(p, c), \text{pkt}, r) = \text{Leaf}(p', c)} \quad \frac{\forall 1 \leq i \leq |qs|. \text{push}(qs[i], \text{pkt}, r) = qs'[i]}{\text{push}(\text{Internal}(qs), \text{pkt}, r) = \text{Internal}(qs')} \quad \frac{\mathbf{flow}(\text{pkt}) \neq c}{\text{push}(\text{Leaf}(p, c), \text{pkt}, r) = \text{Leaf}(p, c)}$$

Informally, we recursively push to all subtrees but only the PIFOs on leaves with the packet's flow are updated.

Definition 1.4. For $t \in \mathbf{Topo}$, define $\mathbf{pop} : \mathbf{RioTree}(t) \times \mathbf{OrdTree}(t) \rightarrow \mathbf{Pkt} \times \mathbf{RioTree}(t)$ such that

$$\frac{\text{pop}(p) = (\text{pkt}, p')}{\text{pop}(\text{Leaf}(p, c), \text{Leaf}) = (\text{pkt}, \text{Leaf}(p', c))} \quad \frac{\text{pop}(qs[i], os[i]) = (\text{pkt}, q) \quad \forall 1 \leq j \leq |qs|. j \neq i \wedge \text{pop}(qs[j], os[j]) = (\text{pkt}', q') \implies rs[i] < rs[j]}{\text{pop}(\text{Internal}(qs), \text{Internal}(rs, os)) = (\text{pkt}, \text{Internal}(qs[q/i]))}$$

Informally, we recursively pop the smallest ranked, poppable subtree.

2 Modelling Scheduling Algorithms

Like [MLF+23], we model scheduling algorithms through a *control*, a thin-layer over the tree policies run on. They keep track of

- (1) a state from a fixed collection **St**
- (2) an underlying tree of buffered packets
- (3) scheduling transactions that update state and construct auxillary structures to push or pop the tree

They come in two flavors: Rio & PIFO Controls. The former determines the order to forward packets at dequeue while latter does so at enqueue.

2.1 Controls

$$\begin{array}{c}
 \frac{z_{\text{pre-push}}(s, \text{pkt}) = (r, s') \quad \text{push}(q, \text{pkt}, r) = q'}{\text{push}((s, q, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}}), \text{pkt}) = (s', q', z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}})} \text{RioCtrl-Push} \\
 \frac{z_{\text{pre-pop}}(s) = (o, s') \quad \text{pop}(q, o) = (\text{pkt}, q') \quad z_{\text{post-pop}}(s', \text{pkt}) = s''}{\text{pop}((s, q, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}})) = (\text{pkt}, (s'', q', z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}}))} \text{RioCtrl-Pop} \\
 \frac{z_{\text{pre-push}}(s, \text{pkt}) = (pt, s') \quad \text{push}(q, \text{pkt}, pt) = q'}{\text{push}((s, q, z_{\text{pre-push}}, z_{\text{post-pop}}), \text{pkt}) = (s', q', z_{\text{pre-push}}, z_{\text{post-pop}})} \text{PIFOCtrl-Push} \\
 \frac{\text{pop}(q) = (\text{pkt}, q') \quad z_{\text{post-pop}}(s, \text{pkt}) = s'}{\text{pop}((s, q, z_{\text{pre-push}}, z_{\text{post-pop}})) = (\text{pkt}, (s', q', z_{\text{pre-push}}, z_{\text{post-pop}}))} \text{PIFOCtrl-Pop}
 \end{array}$$

Figure 2. Pushing and Popping Controls

Definition 2.1. Let $t \in \mathbf{Topo}$ and *scheduling transactions* $z_{\text{pre-push}}$ and $z_{\text{post-pop}}$ be partial functions

$$\begin{aligned}
 z_{\text{pre-push}} &: \mathbf{St} \times \mathbf{Pkt} \rightarrow \mathbf{Path}(t) \times \mathbf{St} \\
 z_{\text{post-pop}} &: \mathbf{St} \times \mathbf{Pkt} \rightarrow \mathbf{St}
 \end{aligned}$$

Define $\mathbf{PIFOControl}(t, z_{\text{pre-push}}, z_{\text{post-pop}})$ to be the set of quadruples

$$(s, q, z_{\text{pre-push}}, z_{\text{post-pop}})$$

where $s \in \mathbf{St}$ and well-formed $q \in \mathbf{PIFOTree}(t)$.

Definition 2.2. Let $t \in \mathbf{Topo}$ and *scheduling transactions* $z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}}$ be partial functions

$$\begin{aligned}
 z_{\text{pre-push}} &: \mathbf{St} \times \mathbf{Pkt} \rightarrow \mathbf{Rk} \times \mathbf{St} \\
 z_{\text{pre-pop}} &: \mathbf{St} \rightarrow \mathbf{OrdTree}(t) \times \mathbf{St} \\
 z_{\text{post-pop}} &: \mathbf{St} \times \mathbf{Pkt} \rightarrow \mathbf{St}
 \end{aligned}$$

Define $\mathbf{RioControl}(t, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}})$ to be the set of quintuples

$$(s, q, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}})$$

where $s \in \mathbf{St}$ and $q \in \mathbf{RioTree}(t)$.

Both controls admit push and pop operations:

$$\begin{aligned}
 \text{push} &: \mathbf{PIFOControl}(t, z_{\text{pre-push}}, z_{\text{post-pop}}) \times \mathbf{Pkt} \rightarrow \mathbf{PIFOControl}(t, z_{\text{pre-push}}, z_{\text{post-pop}}) \\
 \text{pop} &: \mathbf{PIFOControl}(t, z_{\text{pre-push}}, z_{\text{post-pop}}) \rightarrow \mathbf{Pkt} \times \mathbf{PIFOControl}(t, z_{\text{pre-push}}, z_{\text{post-pop}}) \\
 \text{push} &: \mathbf{RioControl}(t, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}}) \times \mathbf{Pkt} \rightarrow \mathbf{RioControl}(t, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}}) \\
 \text{pop} &: \mathbf{RioControl}(t, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}}) \rightarrow \mathbf{Pkt} \times \mathbf{RioControl}(t, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}})
 \end{aligned}$$

Their semantics are written out in full in Figure 2.

2.2 Simulations

Definition 2.3. For $t \in \mathbf{Topo}$, define $\text{empty}_t \in \mathbf{PIFOTree}(t)$ such that

$$\frac{p \in \mathbf{PIFO}(\mathbf{Pkt}) \quad \text{pop}(p) \text{ is undefined}}{\text{empty}_* = \text{Leaf}(p)} \quad \frac{ts \in \mathbf{Topo}^n \quad p \in \mathbf{PIFO}(\{1, \dots, n\}) \quad \forall 1 \leq i \leq n. qs[i] = \text{empty}_{ts[i]}}{\text{empty}_{\text{Node}(ts)} = \text{Internal}(p, qs)}$$

Informally, empty_t is a PIFO tree of topology t , with empty PIFOs at all nodes.

Definition 2.4. Define relation $\rightarrow \subseteq \mathbf{PIFOControl}(t, Z_{\text{pre-push}}, Z_{\text{post-pop}}) \times \mathbf{PIFOControl}(t, Z_{\text{pre-push}}, Z_{\text{post-pop}})$ by

$$\frac{\text{pkt} \in \mathbf{Pkt} \quad \text{push}(c, \text{pkt}) = c'}{c \rightarrow c'} \text{ Step-Push} \quad \frac{\text{pop}(c) = (\text{pkt}, c')}{c \rightarrow c'} \text{ Step-Pop}$$

i.e. $c \rightarrow c'$ if c' is a push or pop from c . We write \rightarrow^* for the reflexive transitive closure of \rightarrow .

Definition 2.5. A partial function

$$f : \mathbf{PIFOControl}(t, Z_{\text{pre-push}}, Z_{\text{post-pop}}) \rightarrow \mathbf{RioControl}(t', Z'_{\text{pre-push}}, Z'_{\text{pre-pop}}, Z'_{\text{post-pop}})$$

and control $c_{\text{init}} \in \text{dom } f$ form a *simulation* if the following conditions are satisfied:

- (1) The PIFO tree in c_{init} is empty, i.e. $c_{\text{init}} = (s, \text{empty}_t, Z_{\text{pre-push}}, Z_{\text{post-pop}})$.
- (2) When $c_{\text{init}} \rightarrow^* c_1$ and $f(c_1) = c_2$, we can guarantee the following:

- (a) $\text{pop}(c_1) \text{ is undefined} \implies \text{pop}(c_2) \text{ is undefined}$
- (b) $\text{pop}(c_1) = (\text{pkt}, c'_1) \implies \text{pop}(c_2) = (\text{pkt}, c'_2) \wedge f(c'_1) = c'_2$
- (c) $\text{push}(c_1, \text{pkt}) = c'_1 \implies \text{push}(c_2, \text{pkt}) = c'_2 \wedge f(c'_1) = c'_2$

Definition 2.6. For $t \in \mathbf{Topo}$, define the set $\mathbf{CTopo}(t)$ such that

$$\frac{c \in \mathbf{Class}}{c \in \mathbf{CTopo}(*)} \quad \frac{\forall 1 \leq i \leq |ts|. \tau s[i] \in \mathbf{CTopo}(ts[i])}{\text{Node}(\tau s) \in \mathbf{CTopo}(\text{Node}(ts))}$$

These are topologies decorated with classes at the leaves.

As we usually use t to denote elements of \mathbf{Topo} , we'll use τ for elements of \mathbf{CTopo} .

Definition 2.7. For $t \in \mathbf{Topo}$, define $\text{proj} : \mathbf{CTopo}(t) \times \mathbf{PIFOTree}(t) \rightarrow \mathbf{RioTree}(t)$ such that

$$\frac{}{\text{proj}(c, \text{Leaf}(p)) = \text{Leaf}(c, p)} \quad \frac{\forall 1 \leq i \leq |qs|. \text{proj}(\tau s[i], qs[i]) = qs'[i]}{\text{proj}(\text{Node}(\tau s), \text{Internal}(p, qs)) = \text{Internal}(qs')}$$

3 Example Controls & Simulations

For all further discussion, we make transparent our previously opaque collections:

$$\mathbf{St} = \{s \mid s \text{ is dictionary mapping } \mathbf{string} \rightarrow \mathbf{float}\} \quad \mathbf{Rk} = \mathbf{Class} = \mathbb{N} \quad \mathbf{Im}(\mathbf{flow}) = [0, n-1]$$

for some $n \in \mathbb{N}$.

3.1 Round-Robin

```

1 def z_pre-push(st, pkt):
2     r = arrival_time(pkt)
3     f = flow(pkt)
4     rank_ptr = "r_" + str(f)
5     r_i = int(st[rank_ptr])
6     st["rank_ptr"] += float(n)
7     return ((f, r_i) :: r, st)

1 def z_post-pop(st, pkt):
2     f = flow(pkt)
3     turn = int(s["turn"])
4     i = turn
5     while i != f:
6         st["r_" + str(i)] += n
7         i = (i + 1) % n
8     st["turn"] = float((f + 1) % n)
9     if turn >= st["turn"]:
10         st["cycle"] += 1
11     return st

```

(a) Round-Robin PIFO Control

```

1 def z_pre-push(st, pkt):
2     return arrival_time(pkt)

1 def z_pre-pop(st):
2     turn = int(s["turn"])
3     rs = []
4     for i in range(n):
5         # Python %, e.g. -2 % 3 == 1, not -2
6         rs.append((i - turn) % n)
7     return Internal(rs, [Leaf] * n)

1 def z_post-pop(st, pkt):
2     f = flow(pkt)
3     st["turn"] = float((f + 1) % n)
4     return st

```

(b) Round-Robin Rio Control

Figure 3. Scheduling Transactions

Let's put our theory to use by constructing PIFO and Rio controls for $\mathbf{rr}[(\mathbf{FIFO}[0], \mathbf{FIFO}[1], \dots, \mathbf{FIFO}[n-1])]$. Both controls use the same underlying topologies, namely

$$t = \text{Node}(\underbrace{*, *, \dots, *}_{n \text{ times}}) \in \mathbf{Topo} \quad \tau = \text{Node}((*, 0), (*, 1), \dots, (*, n-1)) \in \mathbf{CTopo}$$

Figure 3 describes their scheduling transactions in pseudocode. Therefore, we have the materials to define

$$\mathbf{PIFOControl}(t, z_{\text{pre-push}}, z_{\text{post-pop}}) \quad \text{and} \quad \mathbf{RioControl}(t, z'_{\text{pre-push}}, z'_{\text{pre-pop}}, z'_{\text{post-pop}})$$

i.e. the collection of PIFO and Rio controls for our program. Let's find a simulation between them!

Definition 3.1. Let $c_{\text{RR}} = (s, \text{empty}_t, z_{\text{pre-push}}, z_{\text{post-pop}}) \in \mathbf{PIFOControl}(t, z_{\text{pre-push}}, z_{\text{post-pop}})$, where

$$s["turn"] = 0 \quad s["cycle"] = 1 \quad s["r_" + \text{str}(i)] = i$$

for all $0 \leq i \leq n-1$

Definition 3.2. For set S , define ranks : $\mathbf{PIFO}(S) \times S \rightarrow \mathcal{M}(\mathbf{Rk})$ ¹ such that

$$\begin{array}{c} \frac{\text{pop}(p') = (j, p) \quad m = \min(\text{ranks}(p', j)) \quad i = j}{\text{ranks}(p, i) = \text{ranks}(p', i) - \{m\}} \quad \frac{\text{pop}(p') = (j, p) \quad i \neq j}{\text{ranks}(p) = \text{ranks}(p', i)} \\ \frac{\text{push}(p', j, r) = p \quad i = j}{\text{ranks}(p, i) = \text{ranks}(p', i) + \{r\}} \quad \frac{\text{pop}(p) \text{ is undefined}}{\text{ranks}(p, i) = \{\}} \quad \frac{\text{push}(p', j, r) = p \quad i \neq j}{\text{ranks}(p, i) = \text{ranks}(p', i)} \end{array}$$

Informally, $\text{ranks}(p, i)$ is the multiset of ranks with which i lives in PIFO p .

¹We use $\mathcal{M}(X)$ to denote the collection of **multisets** with entries in set X .

Lemma 3.3. If $c_{RR} \rightarrow^* c = (s, q, z_{\text{pre-push}}, z_{\text{post-pop}})$, then both $z_{\text{pre-push}}$ and $z_{\text{post-pop}}$ are defined on $\{\text{pkt} \in \mathbf{Pkt} \mid (s, \text{pkt})\}$

Proof Sketch. This follows by induction on \rightarrow^* and the following observations:

- $z_{\text{pre-push}}$ and $z_{\text{post-pop}}$ are only undefined if they access unbound keys in the state.
- All keys in s accessed by $z_{\text{pre-push}}$ and $z_{\text{post-pop}}$ are defined in c_{RR} 's state.
- Neither $z_{\text{pre-push}}$ nor $z_{\text{post-pop}}$ unbind keys in the state, and, therefore, if

$$(s, q, z_{\text{pre-push}}, z_{\text{post-pop}}) \rightarrow (s', q', z_{\text{pre-push}}, z_{\text{post-pop}})$$

all keys bound in s are bound s' . □

Lemma 3.4. If $c_{RR} \rightarrow^* c = (s, q, z_{\text{pre-push}}, z_{\text{post-pop}})$ and p is q 's root PIFO, i.e. $q = \text{Internal}(p, qs)$, then

$$\text{ranks}(p, i) = \begin{cases} \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s[\text{"cycle"}] \leq x < s[\text{"r_"} + \text{str}(i)] \right\} & i < s[\text{"turn"}] \\ \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s[\text{"cycle"}] - 1) \leq x < s[\text{"r_"} + \text{str}(i)] \right\} & i \geq s[\text{"turn"}] \end{cases}$$

for all $0 \leq i \leq n - 1$.

Proof Sketch. This follows by induction on \rightarrow^* and careful analysis of lines 5-7 of $z_{\text{post-pop}}$ in Figure 3a. □

See Section A for a complete proofs of these lemmas.

Theorem 3.5. Consider the function

$$f : \mathbf{PIFOControl}(t, z_{\text{pre-push}}, z_{\text{post-pop}}) \rightarrow \mathbf{RioControl}(t, z'_{\text{pre-push}}, z'_{\text{pre-pop}}, z'_{\text{post-pop}})$$

such that $f(s, q, z_{\text{pre-push}}, z_{\text{post-pop}}) = (s', \text{proj}(\tau, q), z'_{\text{pre-push}}, z'_{\text{pre-pop}}, z'_{\text{post-pop}})$, where

$$s'[x] = \begin{cases} s[x] & x = \text{"turn"} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Together f and c_{RR} form a simulation.

Proof. We'll verify each condition of Definition 2.5 one by one.

- (1) By Definition 3.1, the PIFO tree in c_{RR} is empty.
- (2) Let $c_{RR} \rightarrow^* c_1 = (s_1, q_1, z_{\text{pre-push}}, z_{\text{pre-pop}})$ and $f(c_1) = c_2 = (s_2, q_2, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}})$.
 - (a) Suppose $\text{pop}(c_1)$ is undefined. By Lemma 3.3, $z_{\text{post-pop}}$ is defined on (s_1, pkt) for all $\text{pkt} \in \mathbf{Pkt}$. Therefore, by rule PIFOCtrl-Pop in Figure 2, $\text{pop}(q_1)$ must be undefined. Since q_1 is well-formed, all PIFOs on it are empty. Similarly, all PIFOs on $q_2 = \text{proj}(\tau, q_1)$ are empty. Hence, $\text{pop}(q_2, o)$ is undefined for all $o \in \mathbf{OrdTree}(t)$, i.e. $\text{pop}(c_2)$ is undefined per rule RioCtrl-Pop in Figure 2.
 - (b)
 - (c) Suppose $\text{push}(c_1, \text{pkt}) = c'_1$. By Lemma 3.3, $z'_{\text{pre-push}}$ is defined on (s_2, pkt) for all $\text{pkt} \in \mathbf{Pkt}$. From inspecting Definition 1.3, $\text{push}(q, \text{pkt}, r)$ is defined for all $q \in \mathbf{RioTree}(t)$, $\text{pkt} \in \mathbf{Pkt}$, and $r \in \mathbf{Rk}$. Therefore, by rule RioCtrl-Push in Figure 2, $\text{push}(c_2, \text{pkt}) = c'_2$ is defined.

□

3.2 Strict

References

- [MLF⁺23] Anshuman Mohan, Yunhe Liu, Nate Foster, Tobias Kappé, and Dexter Kozen. Formal abstractions for packet scheduling, 2023.

A Proofs for Section 3

Lemma 3.4. If $c_{RR} \rightarrow^* c = (s, q, z_{\text{pre-push}}, z_{\text{post-pop}})$ and p is q 's root PIFO, i.e. $q = \text{Internal}(p, qs)$, then

$$\text{ranks}(p, i) = \begin{cases} \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s[\text{"cycle"}] \leq x < s[\text{"r_"} + \text{str}(i)]\} & i < s[\text{"turn"}] \\ \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s[\text{"cycle"}] - 1) \leq x < s[\text{"r_"} + \text{str}(i)]\} & i \geq s[\text{"turn"}] \end{cases}$$

for all $0 \leq i \leq n - 1$.

Proof. We'll proceed by induction on \rightarrow^* .

(Base Case) Let $c = c_{RR}$ and fix arbitrary $i \in [0, n - 1]$.

Recall $q = \text{empty}_t$ by Definition 3.1. Therefore, $\text{pop}(p)$ is undefined by Definition 2.3. Hence,

$$\text{ranks}(p, i) = \{\}$$

by Definition 3.2. Definition 3.1 also insists

$$s[\text{"turn"}] = 0 \quad s[\text{"cycle"}] = 1 \quad s[\text{"r_"} + \text{str}(i)] = i$$

Hence, $i \geq s[\text{"turn"}]$ and

$$\begin{aligned} &= \begin{cases} \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s[\text{"cycle"}] \leq x < s[\text{"r_"} + \text{str}(i)]\} & i < s[\text{"turn"}] \\ \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s[\text{"cycle"}] - 1) \leq x < s[\text{"r_"} + \text{str}(i)]\} & i \geq s[\text{"turn"}] \end{cases} \\ &= \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s[\text{"cycle"}] - 1) \leq x < s[\text{"r_"} + \text{str}(i)]\} \\ &= \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge 0 \leq x < i\} = \{\} \end{aligned}$$

Both sides of the desired equality are therefore the empty multiset $\{\}$.

(Inductive Step) Let $c' = (s', q', z_{\text{pre-push}}, z_{\text{post-pop}}) \rightarrow c$ and $q' = \text{Internal}(p', qs')$. We have two cases.

(Step-Push) Suppose $c = \text{push}(c', \text{pkt})$ and $f = \text{flow}(\text{pkt})$. From inspecting $z_{\text{pre-push}}$ in Figure 3a,

$$\text{push}(p', f, s'[\text{"r_"} + \text{str}(f)]) = p \quad s[x] = \begin{cases} s'[x] + n & x = \text{"r_"} + \text{str}(f) \\ s'[x] & \text{otherwise} \end{cases}$$

Hence, by Definition 3.2 and the IH, for $i \neq f$,

$$\begin{aligned} &= \text{ranks}(p, i) = \text{ranks}(p', i) \\ &= \begin{cases} \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s'[\text{"cycle"}] \leq x < s'[\text{"r_"} + \text{str}(i)]\} & i < s'[\text{"turn"}] \\ \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s'[\text{"cycle"}] - 1) \leq x < s'[\text{"r_"} + \text{str}(i)]\} & i \geq s'[\text{"turn"}] \end{cases} \\ &= \begin{cases} \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s[\text{"cycle"}] \leq x < s[\text{"r_"} + \text{str}(i)]\} & i < s[\text{"turn"}] \\ \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s[\text{"cycle"}] - 1) \leq x < s[\text{"r_"} + \text{str}(i)]\} & i \geq s[\text{"turn"}] \end{cases} \end{aligned}$$

Instead, for $i = f$, let $r = s'[\text{"r_"} + \text{str}(i)]$. Definition 3.2 and the IH then once again show

$$\begin{aligned} &= \text{ranks}(p, i) = \text{ranks}(p', i) + \{r\} \\ &= \begin{cases} \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s'[\text{"cycle"}] \leq x < s'[\text{"r_"} + \text{str}(i)]\} + \{r\} & i < s'[\text{"turn"}] \\ \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s'[\text{"cycle"}] - 1) \leq x < s'[\text{"r_"} + \text{str}(i)]\} + \{r\} & i \geq s'[\text{"turn"}] \end{cases} \\ &= \begin{cases} \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s'[\text{"cycle"}] \leq x < s'[\text{"r_"} + \text{str}(i)] + n\} & i < s'[\text{"turn"}] \\ \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s'[\text{"cycle"}] - 1) \leq x < s'[\text{"r_"} + \text{str}(i)] + n\} & i \geq s'[\text{"turn"}] \end{cases} \\ &= \begin{cases} \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s[\text{"cycle"}] \leq x < s[\text{"r_"} + \text{str}(i)]\} & i < s[\text{"turn"}] \\ \{x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s[\text{"cycle"}] - 1) \leq x < s[\text{"r_"} + \text{str}(i)]\} & i \geq s[\text{"turn"}] \end{cases} \end{aligned}$$

(Step-Pop) Suppose $(\text{pkt}, c) = \text{pop}(c')$ and $f = \text{flow}(\text{pkt})$. From inspecting $z_{\text{post-pop}}$ in Figure 3a,

$$\text{pop}(p') = (f, p) \quad s[\text{"cycle"}] \in \{s'[\text{"cycle"}], s'[\text{"cycle"}] + 1\}$$

Consider the case where $s[\text{"cycle"}] = s'[\text{"cycle"}]$, i.e. $s'[\text{"turn"}] \leq f < f + 1 = s[\text{"turn"}]$.

- For $i < s'[\text{"turn"}]$ or $i \geq s[\text{"turn"}]$, lines 5-7 of $z_{\text{post-pop}}$ in Figure 3a show

$$s[\text{"r_"} + \text{str}(i)] = s'[\text{"r_"} + \text{str}(i)]$$

Therefore, by Definition 3.2 and the IH,

$$\begin{aligned} &= \text{ranks}(p, i) = \text{ranks}(p', i) \\ &= \begin{cases} \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s'[\text{"cycle"}] \leq x < s'[\text{"r_"} + \text{str}(i)] \right\} & i < s'[\text{"turn"}] \\ \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s'[\text{"cycle"}] - 1) \leq x < s'[\text{"r_"} + \text{str}(i)] \right\} & i \geq s'[\text{"turn"}] \end{cases} \\ &= \begin{cases} \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s[\text{"cycle"}] \leq x < s[\text{"r_"} + \text{str}(i)] \right\} & i < s[\text{"turn"}] \\ \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s[\text{"cycle"}] - 1) \leq x < s[\text{"r_"} + \text{str}(i)] \right\} & i \geq s[\text{"turn"}] \end{cases} \end{aligned}$$

- Suppose $s'[\text{"turn"}] \leq i < f$. By the IH,

$$\text{ranks}(p', j) = \left\{ x \in \mathbf{Rk} \mid x \equiv j \pmod{n} \wedge n \cdot (s'[\text{"cycle"}] - 1) \leq x < s'[\text{"r_"} + \text{str}(j)] \right\}$$

Since popping p' returned f , it must be a minimally ranked element. However,

$$n \cdot (s'[\text{"cycle"}] - 1) + i < n \cdot (s'[\text{"cycle"}] - 1) + f$$

Therefore, $\text{ranks}(p', i) = \{\}$: i.e. there are no numbers congruent to $i \pmod{n}$ in

$$\left[n \cdot (s'[\text{"cycle"}] - 1), s'[\text{"r_"} + \text{str}(i)] \right)$$

The same is therefore true of

$$\left[n \cdot s'[\text{"cycle"}], s'[\text{"r_"} + \text{str}(i)] + n \right)$$

Since lines 5-7 of $z_{\text{post-pop}}$ in Figure 3a show $s[\text{"r_"} + \text{str}(i)] = s'[\text{"r_"} + \text{str}(i)] + n$,

$$\begin{aligned} &(\text{By Definition 3.2}) \quad = \text{ranks}(p, i) = \text{ranks}(p', i) \\ &= \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s'[\text{"cycle"}] - 1) \leq x < s'[\text{"r_"} + \text{str}(i)] \right\} \\ &= \{\} \\ &= \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s'[\text{"cycle"}] \leq x < s'[\text{"r_"} + \text{str}(i)] + n \right\} \\ &= \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s[\text{"cycle"}] \leq x < s[\text{"r_"} + \text{str}(i)] \right\} \end{aligned}$$

- For $i = f$, lines 5-7 of $z_{\text{post-pop}}$ in Figure 3a yet again show

$$s[\text{"r_"} + \text{str}(i)] = s'[\text{"r_"} + \text{str}(i)]$$

Therefore, by Definition 3.2 and the IH,

$$\begin{aligned} &= \text{ranks}(p, i) = \text{ranks}(p') - \{n \cdot (s'[\text{"cycle"}] - 1)\} \\ &= \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot (s'[\text{"cycle"}] - 1) \leq x < s'[\text{"r_"} + \text{str}(i)] \right\} - \{n \cdot (s'[\text{"cycle"}] - 1)\} \\ &= \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s'[\text{"cycle"}] \leq x < s'[\text{"r_"} + \text{str}(i)] \right\} \\ &= \left\{ x \in \mathbf{Rk} \mid x \equiv i \pmod{n} \wedge n \cdot s[\text{"cycle"}] \leq x < s[\text{"r_"} + \text{str}(i)] \right\} \end{aligned}$$

We leave the cases where $s[\text{"cycle"}] = s'[\text{"cycle"}] + 1$ as an exercise as they're much of the same:

- (i) $i \in [s[\text{"turn"}], s'[\text{"turn"}])$ ("r_" + str(i) is unchanged)
- (ii) $i \in [s'[\text{"turn"}], n] \cup [0, f)$ ("r_" + str(i) is incremented by n)
- (iii) $i = f$

□