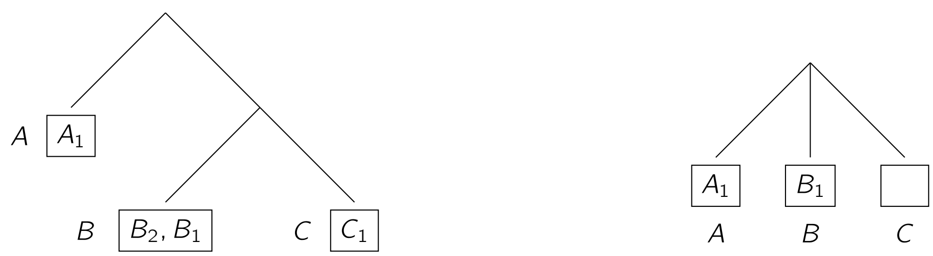
Dequeue-Side Semantics

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

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

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

1 Structure and Semantics of Rio Trees



(a) Over topology Node($*$, Node($*$, $*$))

(b) Over topology Node($*$, $*$, $*$)

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

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

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

$$\frac{}{\mathrm{Leaf} \in \mathbf{OrdTree}(*)} \qquad \frac{\begin{array}{c} ts \in \mathbf{Topo}^n \qquad rs \in \mathbf{Rk}^n \\ \forall 1 \le i < j \le n.\ rs[i] \ne rs[j] \qquad \forall 1 \le i \le n.\ os[i] \in \mathbf{OrdTree}(ts[i]) \end{array}}{\mathrm{Internal}(rs, os) \in \mathbf{OrdTree}(\mathrm{Node}(ts))}$$

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

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

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

$$\frac{\mathbf{flow}(\mathrm{pkt}) = c \qquad \mathrm{push}(p, \mathrm{pkt}, r) = p'}{\mathrm{push}(\mathrm{Leaf}(p, c), \mathrm{pkt}, r) = \mathrm{Leaf}(p', c)} \qquad \frac{\forall 1 \le i \le |qs|.\ \mathrm{push}(qs[i], \mathrm{pkt}, r) = qs'[i]}{\mathrm{push}(\mathrm{Internal}(qs), \mathrm{pkt}, r) = \mathrm{Internal}(qs')} \qquad \frac{\mathbf{flow}(\mathrm{pkt}) \ne c}{\mathrm{push}(\mathrm{Leaf}(p, c), \mathrm{pkt}, r) = \mathrm{Leaf}(p, c)}$$

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 pop : $\mathbf{RioTree}(t) \times \mathbf{OrdTree}(t) \rightharpoonup \mathbf{Pkt} \times \mathbf{RioTree}(t)$ such that

$$\frac{\mathrm{pop}(p) = (\mathrm{pkt}, p')}{\mathrm{pop}(\mathrm{Leaf}(p, c), \mathrm{Leaf}) = (\mathrm{pkt}, \mathrm{Leaf}(p', c))} \qquad \frac{\begin{array}{c} \mathrm{pop}(qs[i], os[i]) = (\mathrm{pkt}, q) \\ \forall 1 \le j \le |qs|.\ j \ne i \wedge \mathrm{pop}(qs[j], os[j]) = (\mathrm{pkt}', q') \implies rs[i] < rs[j] \end{array}}{\mathrm{pop}(\mathrm{Internal}(qs), \mathrm{Internal}(rs, os)) = (\mathrm{pkt}, \mathrm{Internal}(qs[q/i]))}$$

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

$$\dfrac{z_{\text{pre-push}}(s, \mathsf{pkt}) = (r, s') \qquad \mathsf{push}(q, \mathsf{pkt}, r) = q'}{\mathsf{push}((s, q, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}}), \mathsf{pkt}) = (s', q', z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}})}\ \textsf{RioCtrl-Push}$$

$$\dfrac{z_{\text{pre-pop}}(s) = (o, s') \qquad \mathsf{pop}(q, o) = (\mathsf{pkt}, q') \qquad z_{\text{post-pop}}(s', \mathsf{pkt}) = s''}{\mathsf{pop}((s, q, z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}})) = (\mathsf{pkt}, (s'', q', z_{\text{pre-push}}, z_{\text{pre-pop}}, z_{\text{post-pop}}))}\ \textsf{RioCtrl-Pop}$$

$$\dfrac{z_{\text{pre-push}}(s, \mathsf{pkt}) = (pt, s') \qquad \mathsf{push}(q, \mathsf{pkt}, pt) = q'}{\mathsf{push}((s, q, z_{\text{pre-push}}, z_{\text{post-pop}}), \mathsf{pkt}) = (s', q', z_{\text{pre-push}}, z_{\text{post-pop}})}\ \textsf{PIFOCtrl-Push}$$

$$\dfrac{\mathsf{pop}(q) = (\mathsf{pkt}, q') \qquad z_{\text{post-pop}}(s, \mathsf{pkt}) = s'}{\mathsf{pop}((s, q, z_{\text{pre-push}}, z_{\text{post-pop}})) = (\mathsf{pkt}, (s', q', z_{\text{pre-push}}, z_{\text{post-pop}}))}\ \textsf{PIFOCtrl-Pop}$$

Figure 2. Pushing and Popping Controls

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

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

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

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

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

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

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

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

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

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

Both controls admit push and pop operations:

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

Their semantics are written out in full in Figure 2.

2.2 Simulations

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

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

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

Definition 2.4. Define the $\rightarrow \subseteq \textbf{PIFOControl}(t, \text{z}_{\text{pre-push}}, \text{z}_{\text{post-pop}}) \times \textbf{PIFOControl}(t, \text{z}_{\text{pre-push}}, \text{z}_{\text{post-pop}})$ by

$$\frac{\text{pkt} \in \textbf{Pkt} \qquad \text{push}(c, \text{pkt}) = c'}{c \rightarrow c'} \qquad\qquad \frac{\text{pop}(c) = (\text{pkt}, c')}{c \rightarrow c'}$$

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

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

is *simulation* if the following conditions are satisfied:

(1) There exists $c_{init} = (s, q, \text{z}_{\text{pre-push}}, \text{z}_{\text{post-pop}})$ where $q = \text{empty}_t$ and $c_{init} \in \text{dom}\, f$.

(2) When $c_{\text{init}} \rightarrow^* c_1$ and $f(c_1) = c_2$, we can guarantee the following:

$$\text{pop}(c_1) \text{ is undefined} \implies \text{pop}(c_2) \text{ is undefined} \tag{a}$$

$$\text{pop}(c_1) = (\text{pkt}, c_1') \implies \text{pop}(c_2) = (\text{pkt}, c_2') \wedge f(c_1') = c_2' \tag{b}$$

$$\text{push}(c_1, \text{pkt}) = c_1' \implies \text{push}(c_2, \text{pkt}) = c_2' \wedge f(c_1') = c_2' \tag{c}$$

3 Enqueue and Dequeue-Side Equivalence

All further discussion takes **St** = set of dictionaries mapping **string** → **float** and **Rk** = **Class** = $\mathbb{N}$.

3.1 Round-Robin

```
def z_pre-push(st, pkt):
    r = st["counter"]
    st["counter"] += 1
    f = flow(pkt)
    rank_ptr = "r_" + str(f)
    r_i = int(st[rank_ptr])
    st["rank_ptr"] += float(n)
    return ((f, r_i) :: r, st)
```

```
def z_post-pop(st, pkt):
    f = flow(pkt)
    turn = int(s["turn"])
    i = turn
    while i != f:
        st["r_" + str(i)] += n
        i = (i + 1) % n
    st["turn"] = float((f + 1) % n)
    if turn >= st["turn"]:
        st["cycle"] += 1
    return st
```

(a) PIFO Control

```
def z_pre-push(st, pkt):
    r = int(st["counter"])
    st["counter"] += 1.0
    return r
```

```
def z_pre-pop(st):
    turn = int(s["turn"])
    rs = []
    for i in range(n):
        rs.append((i - turn) % n)
    return Internal(rs, [Leaf] * n)
```

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

(b) Rio Control

Figure 3. Scheduling Transactions

For $n \in \mathbb{N}$, let's put our theory to use by constructing PIFO and Rio controls for

$$\mathbf{rr}[(\mathbf{FIFO}[0], \mathbf{FIFO}[1], \ldots, \mathbf{FIFO}[n-1])]$$

Both controls use the same underlying topology, namely

$$t = \text{Node}(\underbrace{*, *, \ldots, *}_{n \text{ times}})$$

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

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

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

References

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