PIEO Trees for Fun and Profit

We assume familiarity with [MLF+23], adopt its notational conventions, and borrow many of its definitions!

1 Structure & Semantics

Definition 1.1. For sets S, D, and predicates F over D, let $\mathbf{PIEO}(S, D, F)$ denote the set of *PIEO*s that

(1) hold entries in S, decorated with meta-data in D
(2) are ordered by **Rk**
(3) support predicates in F
(4) admit partial functions

$$\begin{aligned}
\text{pop} &: \mathbf{PIEO}(S, D, F) \times F \rightharpoonup S \times \mathbf{PIEO}(S, D, F) \\
\text{push} &: \mathbf{PIEO}(S, D, F) \times S \times D \times \mathbf{Rk} \to \mathbf{PIEO}(S, D, F) \\
\text{proj} &: \mathbf{PIEO}(S, D, F) \times F \to \mathbf{PIFO}(S)
\end{aligned}$$

Maps push and pop are as usual. The *projection* $\text{proj}(p, f)$ is the PIFO of entries in p with data satisfying f. These three maps play nicely together:

$$\text{pop}(p, f) \text{ is undefined} \iff \text{pop}(\text{proj}(p, f)) \text{ is undefined} \tag{1}$$

$$\text{pop}(p, f) = (\text{pkt}, p') \implies \text{pop}(\text{proj}(p, f)) = (\text{pkt}, \text{proj}(p', f)) \tag{2}$$

$$\text{proj}(\text{push}(p, s, d, r), f) = \begin{cases} \text{push}(\text{proj}(p, f), s, r) & f(d) \text{ holds true} \\ \text{proj}(p, f) & \text{otherwise} \end{cases} \tag{3}$$

We consider PIEOs p, p' equal if, for all $f \in F$, $\text{proj}(p, f) = \text{proj}(p', f)$, i.e. their projections are always equal. For PIEO p, entry $s \in S$, and predicate $f \in F$, we write

(1) $|p|$ for the number of entries in p
(2) $|p|_s$ for the number of times s occurs in p
(3) $|p|_{s,f}$ for the number of times s occurs in p with associated $d \in D$ such that $f(d)$ holds

We fix an opaque set **Data** and a collection $\mathcal{F}$ of predicates defined on it. These predicates come with a total order $\leq$ and the property that, $\forall d \in \mathbf{Data}$ and $f, f' \in \mathcal{F}$, $f \leq f' \wedge f(d) \implies f'(d)$.

Definition 1.2. The set of *PIEO trees* over $t \in \mathbf{Topo}$, denoted $\mathbf{PIEOTree}(t)$, is defined inductively by

$$\frac{p \in \mathbf{PIEO}(\mathbf{Pkt}, \mathbf{Data}, \mathcal{F})}{\text{Leaf}(p) \in \mathbf{PIEOTree}(*)} \qquad \frac{n \in \mathbb{N} \qquad ts \in \mathbf{Topo}^n \qquad p \in \mathbf{PIEO}(\{1, \dots, n\}, \mathbf{Data}, \mathcal{F}) \qquad \forall i \in [1, n].\ qs[i] \in \mathbf{PIEOTree}(ts[i])}{\text{Internal}(qs, p) \in \mathbf{PIEO}(\text{Node}(ts))}$$

Definition 1.3. Define $\text{pop} : \mathbf{PIEOTree}(t) \times \mathcal{F} \rightharpoonup \mathbf{Pkt} \times \mathbf{PIEOTree}(t)$ by

$$\frac{\text{pop}(p, f) = (\text{pkt}, p')}{\text{pop}(\text{Leaf}(p), f) = (\text{pkt}, \text{Leaf}(p'))} \qquad \frac{\text{pop}(p, f) = (i, p') \qquad \text{pop}(qs[i], f) = (\text{pkt}, q')}{\text{pop}(\text{Internal}(qs, p), f) = (\text{pkt}, \text{Internal}(qs[q'/i], p'))}$$

Definition 1.4. Define $\text{push} : \mathbf{PIEOTree}(t) \times \mathbf{Pkt} \times \mathbf{Data} \times \mathbf{Path}(t) \to \mathbf{PIEOTree}(t)$ by

$$\frac{\text{push}(p, \text{pkt}, d, r) = p'}{\text{push}(\text{Leaf}(p), \text{pkt}, d, r) = \text{Leaf}(p')} \qquad \frac{\text{push}(p, i, d, r) = p' \qquad \text{push}(qs[i], \text{pkt}, d, pt) = q'}{\text{push}(\text{Internal}(qs, p), \text{pkt}, d, (i, r) :: pt) = \text{Internal}(qs[q'/i], p')}$$

Definition 1.5. Let $t \in \mathbf{Topo}$. A *control* over t is a triple (s, q, z), where $s \in \text{St}$ is the *current state*, q is a PIEO tree of topology t, and

$$z : \text{St} \times \mathbf{Pkt} \to \mathbf{Data} \times \mathbf{Path}(t) \times \text{St}$$

is a function called the *scheduling transaction*.

Definition 1.6. Define $|\cdot| : \mathbf{PIEOTree}(t) \to \mathbb{N}$ by

$$|\,\mathsf{Leaf}(p)| = |p| \qquad\qquad |\,\mathsf{Internal}(qs, p)| = \sum_{i=1}^{|qs|} |qs[i]|$$

We say that $q \in \mathbf{PIEOTree}(t)$ is *well-formed* w.r.t $f \in \mathcal{F}$, denoted $\vdash_f q$, if it adheres to the following rules.

$$\frac{}{\vdash_f \mathsf{Leaf}(p)} \qquad\qquad \frac{\forall i \in [1, |qs|],\ \vdash_f qs[i] \wedge |p|_{i,f} = |qs[i]|}{\vdash_f \mathsf{Internal}(qs, p)}$$

We say q is well-formed, denoted $\vdash q$, if there exists $f \in \mathcal{F}$ such that, for all $f' \geq f$, $\vdash_{f'} q$.

2 Projection

Definition 2.1. For $f \in \mathcal{F}$, define $\mathsf{proj}_f : \mathbf{PIEOTree}(t) \to \mathbf{PIFOTree}(t)$ by

$$\frac{p' = \mathsf{proj}(p, f)}{\mathsf{proj}_f(\mathsf{Leaf}(p)) = \mathsf{Leaf}(p')} \qquad\qquad \frac{p' = \mathsf{proj}(p, f) \qquad \forall i \in [1, |qs|],\ qs'[i] = \mathsf{proj}_f(qs[i])}{\mathsf{proj}_f(\mathsf{Internal}(qs, p)) = \mathsf{Internal}(qs', p')}$$

Lemma 2.2. *For $q, q' \in \mathbf{PIEOTree}(t)$,*

$$\forall f \in \mathcal{F},\ \mathsf{proj}_f(q) = \mathsf{proj}_f(q') \implies q = q'$$

Proof. Suppose $\mathsf{proj}_f(q) = \mathsf{proj}_f(q')$ for all $f \in \mathcal{F}$. We'll proceed by induction on t to show $q = q'$.

(Leaf) For $t = *$, let $q = \mathsf{Leaf}(p)$ and $q' = \mathsf{Leaf}(p')$. Since

$$\mathsf{proj}_f(q) = \mathsf{Leaf}(\mathsf{proj}(p, f)) = \mathsf{Leaf}(\mathsf{proj}(p', f)) = \mathsf{proj}_f(q')$$

we know $\mathsf{proj}(p, f) = \mathsf{proj}(p', f)$ for all $f \in \mathcal{F}$. By Definition 1.1, $p = p'$ and hence $q = q'$.

(Node) For $t = \mathsf{Node}(ts)$ and $n = |ts|$, let $q = \mathsf{Internal}(qs, p)$ and $q' = \mathsf{Internal}(qs', p)$. Notice

$$\begin{aligned} \mathsf{proj}_f(p) &= \mathsf{proj}_f(p') \\ \mathsf{proj}_f(qs[i]) &= \mathsf{proj}_f(qs'[i]) \qquad (i = 1, \ldots, n) \end{aligned}$$

for all $f \in \mathcal{F}$. Hence, $p = p'$ via Definition 1.1 and $qs = qs'$ by the inductive hypothesis, i.e. $q = q'$.

$\square$

Lemma 2.3. *For $q \in \mathbf{PIEOTree}(t)$ and $f \in \mathcal{F}$, $\mathsf{pop}(q, f)$ is undefined if and only if $\mathsf{pop}(\mathsf{proj}_f(q))$ is undefined.*

Proof. We'll do induction on t.

(Leaf) For $t = *$, let $q = \mathsf{Leaf}(p)$ and $\mathsf{proj}_f(q) = \mathsf{Leaf}(p')$. By Equation (1) in Definition 1.1,

$$\begin{aligned} \mathsf{pop}(q, f) \text{ is undefined} &\iff \mathsf{pop}(p, f) \text{ is undefined} \\ &\iff \mathsf{pop}(p') \text{ is undefined} \\ &\iff \mathsf{pop}(\mathsf{proj}_f(q)) \text{ is undefined} \end{aligned}$$

(Node) For $t = \mathsf{Node}(ts)$, let $q = \mathsf{Internal}(qs, p)$ and $\mathsf{proj}_f(q) = \mathsf{Internal}(qs', p')$. Notice

$$\mathsf{pop}(p, f) \text{ is undefined} \iff \mathsf{pop}(p') \text{ is undefined}$$

by Equation (1) in Definition 1.1 and

$$\mathsf{pop}(qs[i], f) \text{ is undefined} \iff \mathsf{pop}(qs'[i]) \text{ is undefined } \forall i \in [1, |ts|]$$

by the inductive hypothesis. Hence, $\mathsf{pop}(q, f)$ is undefined if and only if $\mathsf{pop}(\mathsf{proj}_f(q))$ is.

$\square$

Lemma 2.4. *For $q \in \mathbf{PIEOTree}(t)$ and $f \in \mathcal{F}$,*

$$\mathsf{pop}(q, f) = (\mathsf{pkt}, q') \implies \mathsf{pop}(\mathsf{proj}_f(q)) = (\mathsf{pkt}, \mathsf{proj}_f(q'))$$

Proof. More induction on t!

(Leaf)

(Node)

□

Lemma 2.5. *For $q \in \mathbf{PIEOTree}(t)$, $\mathrm{pkt} \in \mathbf{Pkt}$, $d \in \mathbf{Data}$, $pt \in \mathbf{Path}(t)$, and $f \in \mathcal{F}$,*

$$\mathrm{proj}_f(\mathrm{push}(q, \mathrm{pkt}, d, pt)) = \begin{cases} \mathrm{push}(\mathrm{proj}_f(q), \mathrm{pkt}, pt) & f(d) \textit{ holds true} \\ \mathrm{proj}_f(q) & \textit{otherwise} \end{cases}$$

Proof. Even more induction on t!

(Leaf) For $t = *$, let $q = \mathrm{Leaf}(p)$ and $pt = r$. By Equation (3) in Definition 1.1.

$$\begin{aligned} \mathrm{proj}_f(\mathrm{push}(q, \mathrm{pkt}, d, pt)) &= \mathrm{Leaf}(\mathrm{proj}(\mathrm{push}(p, \mathrm{pkt}, d, r), f)) \\ &= \begin{cases} \mathrm{Leaf}(\mathrm{push}(\mathrm{proj}(p, f), \mathrm{pkt}, r)) & f(d) \text{ holds true} \\ \mathrm{Leaf}(\mathrm{proj}(p, f)) & \text{otherwise} \end{cases} \\ &= \begin{cases} \mathrm{push}(\mathrm{proj}_f(q), \mathrm{pkt}, pt) & f(d) \text{ holds true} \\ \mathrm{proj}_f(q) & \text{otherwise} \end{cases} \end{aligned}$$

(Node) For $t = \mathrm{Node}(ts)$, let $q = \mathrm{Internal}(qs, p)$ and $pt = (i, r) :: pt'$. Define

$$p' = \mathrm{push}(p, i, d, r) \qquad q' = \mathrm{push}(qs[i]\, \mathrm{pkt}, d, pt') \qquad qs' = qs[q'/i]$$

By Equation (3) in Definition 1.1 and the inductive hypothesis,

$$\mathrm{proj}(p', f) = \begin{cases} \mathrm{push}(\mathrm{proj}(p, f), i, r) & f(d) \text{ holds true} \\ \mathrm{proj}(p, f) & \text{otherwise} \end{cases}$$

$$\mathrm{proj}_f(qs', \mathrm{pkt}, d, pt')) = \begin{cases} \mathrm{push}(\mathrm{proj}_f(qs[i]), \mathrm{pkt}, pt') & f(d) \text{ holds true} \\ \mathrm{proj}_f(qs[i]) & \text{otherwise} \end{cases}$$

Since $\mathrm{push}(q, \mathrm{pkt}, d, pt)) = \mathrm{Internal}(qs', p')$, the desired result follows.

□

3 Embedding & Simulation

Definition 3.1. Let $t_1, t_2 \in \mathbf{Topo}$. We call a relation $R \subseteq \mathbf{PIEOTree}(t_1) \times \mathbf{PIEOTree}(t_2)$ a *simulation* if, for all $\mathrm{pkt} \in \mathbf{Pkt}$, $f \in \mathcal{F}$, and $q_1\ R\ q_2$,

(1) If $\mathrm{pop}(q_1, f)$ is undefined, then so is $\mathrm{pop}(q_2, f)$
(2) If $\mathrm{pop}(q_1, f) = (\mathrm{pkt}, q_1')$, then $\mathrm{pop}(q_2) = (\mathrm{pkt}, q_2')$ such that $q_1'\ R\ q_2'$.
(3) For all $pt_1 \in \mathbf{Path}(t_1)$ and $d \in \mathbf{Data}$, there exists $pt_2 \in \mathbf{Path}(t_2)$ such that

$$\mathrm{push}(q_1, \mathrm{pkt}, d, pt_1)\ R\ \mathrm{push}(q_2, \mathrm{pkt}, d, pt_2)$$

If such a simulation exists, we say that q_1 is *simulated* by q_2, and we write $q_1 \preccurlyeq q_2$.

Remark 3.2. For all further discussion, we assume our embeddings are injective.

Definition 3.3. For $t_1, t_2 \in \mathbf{Topo}$, let f be an embedding from t_1 to t_2. We lift f to a map $\overline{f}$ from $\mathbf{PIEOTree}(t_1)$ to $\mathbf{PIEOTree}(t_2)$ inductively.

- For $t_1 = *$, define $\overline{f}(q) = q$. This is well-defined by [MLF+23, Lemma 5.2].
- For $t_1 = \mathrm{Node}(ts_1)$, $n = |ts_1|$, $q = \mathrm{Internal}(qs, p)$, construct $\overline{f}_\alpha(q) \in \mathbf{PIEOTree}(t_2/\alpha)$ for each prefix α of $f(i)$ for some $i \in [1, n]$. Inductively, we'll build up from $f(i)$'s to ϵ and set $\overline{f}(q) = \overline{f}_\epsilon(q)$.
  - Let $\alpha = f(i)$ for some $i \in [1, n]$. We'll set $\overline{f}_\alpha(q) = \overline{f_i}(qs[i])$, where f_i embeds t_1/i into $t_2/f(i)$ as per [MLF+23, Lemma 5.2]. This well-defined by the injectivity of f.
  - Let α point to a transient node, say with m children. For $1 \leq j \leq m$ such that $\alpha \cdot j$ is not a prefix of some $f(i)$, define $\overline{f}(q)_{\alpha \cdot j}$ to be the PIEO tree with empty PIEOs on all leaves and internal nodes. With this and recursion, we know $\overline{f}(q)_{\alpha \cdot j} \in \mathbf{PIEOTree}(t_2/(\alpha \cdot j))$ for all $j \in [1, m]$.
    We create a new PIEO p_α as follows:
    (1) Start with p_α empty
    (2) For each i in p such that $\alpha \cdot j$ is a prefix of $f(i)$, push j into p_α with i's data and rank

Finally, for all $j \in [1, m]$, set $qs_\alpha[j] = \overline{f}(q)_{\alpha \cdot j}$ and $\overline{f}(q)_\alpha = \mathrm{Internal}(qs_\alpha, p_\alpha)$.

Theorem 3.4. *The following diagram commutes*

$$\begin{tikzcd}
\mathbf{PIEOTree}(t_1) \arrow[r, "\overline{f}"] \arrow[d, "\mathrm{proj}_g"] & \mathbf{PIEOTree}(t_2) \arrow[d, "\mathrm{proj}_g"] \\
\mathbf{PIFOTree}(t_1) \arrow[r, "\widehat{f}"] & \mathbf{PIFOTree}(t_2)
\end{tikzcd}$$

In other words, for $q \in \mathbf{PIEOTree}(t_1)$ and $g \in \mathcal{F}$, $\mathrm{proj}_g(\overline{f}(q)) = \widehat{f}(\mathrm{proj}_g(q))$.

Proof. We'll proceed by induction on t_1. Suppose $t_1 = *$ and $q = \mathrm{Leaf}(p)$. By [MLF+23, Lemma 5.3], $t_2 = *$ as well. By Definition 3.3 and [MLF+23, Definition 5.4], both $\overline{f}$ and $\widehat{f}$ are the identity. Hence,

$$\mathrm{proj}_g(\overline{f}(q)) = \mathrm{proj}_g(q) = \widehat{f}(\mathrm{proj}_g(q))$$

Suppose $t_1 = \mathrm{Node}(ts)$ and $q = \mathrm{Internal}(qs, p)$. For any prefix α of $f(i)$ for $i \in [1, |ts|]$, we'll show

$$\mathrm{proj}_g(\overline{f}(q)_\alpha) = \widehat{f}(\mathrm{proj}_g(q))_\alpha \tag{$*$}$$

by inverse induction on α. Instantiating Equation ($*$) with $\alpha = \epsilon$ yields the desired result.

- For $\alpha = f(i)$, Equation ($*$) holds by the outer inductive hypothesis because

$$\mathrm{proj}_g(\overline{f}(q)_\alpha) = \mathrm{proj}_g(\overline{f_i}(q)) = \widehat{f_i}(\mathrm{proj}_g(q)) = \widehat{f}(\mathrm{proj}_g(q))_\alpha$$

- Suppose α is some strict prefix of $f(i)$, pointing to a node with m children. Let

$$\mathrm{proj}_g(\overline{f}(q)) = \mathrm{Internal}(qs', p') \qquad \text{and} \qquad \widehat{f}(\mathrm{proj}_g(q)) = \mathrm{Internal}(qs'', p'')$$

  There's two parts to showing Equation ($*$), namely $qs' = qs''$ and $p' = p''$.
  - For all $j \in [1, m]$,

$$\mathrm{proj}_g(\overline{f}(q)_{\alpha \cdot j}) = \widehat{f}(\mathrm{proj}_g(q))_{\alpha \cdot j}$$

    For j such that $\alpha \cdot j$ is a prefix of some $f(i)$, this follows from the inner inductive hypothesis. For all other j, notice the LHS and RHS are both PIFO trees of topology t_2, with empty PIFOs on all leaves and internal nodes. Hence, $qs'[j] = qs''[j]$ for all $j \in [1, m]$, i.e. $qs' = qs''$.
  - By inspection, it's clear following the construction for p_α from Definition 3.3 and then computing the projection $\mathrm{proj}(p_\alpha, g)$ yields the same result as following the recipe for p_α from [MLF+23, Definition 5.4] on the projection $\mathrm{proj}_g(q)$: that is, exactly when we filter out elements not satisfying g does not matter. Hence, $p' = p''$.

□

Theorem 3.5. *Let $t_1, t_2 \in \mathbf{Topo}$. If f embeds t_1 into t_2, then*

$$R = \{(q, \overline{f}(q)) \mid q \in \mathbf{PIEOTree}(t_1)\}$$

is a simulation.

Proof. We'll show the conditions from Definition 3.1 hold. Fix $g \in \mathcal{F}$ and $q_1 \in \mathbf{PIEOTree}(t_1)$. Let $q_2 = \overline{f}(q_1)$.

(1) Suppose $\mathrm{pop}(q_1, g)$ is undefined. Applying both Lemma 2.3 and [MLF+23, Lemma 5.6], notice $\mathrm{pop}(\widehat{f}(\mathrm{proj}_g(q_1)))$ is undefined. By Theorem 3.4, $\widehat{f}(\mathrm{proj}_g(q_1)) = \mathrm{proj}_g(q_2)$. Hence, $\mathrm{pop}(\mathrm{proj}_g(q_2))$ is undefined. Applying Lemma 2.3 once more, $\mathrm{pop}(q_2, g)$ is undefined.

(2) Suppose $\mathrm{pop}(q_1, g)$ is defined. By Lemma 2.3 and [MLF+23, Lemma 5.6], $\mathrm{pop}(\widehat{f}(\mathrm{proj}_g(q_1)))$ is defined. Hence, $\mathrm{pop}(\mathrm{proj}_g(q_2))$ is defined via Theorem 3.4. By Lemma 2.3, $\mathrm{pop}(q_2, g)$ is defined. Let's say

$$\mathrm{pop}(q_1, g) = (\mathrm{pkt}_1, q_1') \qquad \mathrm{pop}(q_2, g) = (\mathrm{pkt}_2, q_2')$$

By Lemma 2.4,

$$\mathrm{pop}(\mathrm{proj}_g(q_1)) = (\mathrm{pkt}_1, \mathrm{proj}_g(q_1')) \qquad \mathrm{pop}(\mathrm{proj}_g(q_2)) = (\mathrm{pkt}_2, \mathrm{proj}_g(q_2'))$$

By [MLF+23, Lemma 5.7],

$$\mathsf{pop}(\widehat{f}(\mathsf{proj}_g(q_1))) = (\mathsf{pkt}_1, \widehat{f}(\mathsf{proj}_g(q_1')))$$

By Theorem 3.4,

$$\mathsf{pkt}_1 = \mathsf{pkt}_2$$
$$\widehat{f}(\mathsf{proj}_g(q_1')) = \mathsf{proj}_g(q_2') \tag{†}$$

Since our choice of g was arbitrary, notice Equation (†) holds for all $g \in \mathcal{F}$ (this is not true!). Hence, using Theorem 3.4,

$$\mathsf{proj}_{g'}(\overline{f}(q_1')) = \widehat{f}(\mathsf{proj}_{g'}(q_1')) = \mathsf{proj}_{g'}(q_2')$$

for all $g' \in \mathcal{F}$. At last, Lemma 2.2 yields $\overline{f}(q_1') = q_2'$.

(3) Consider $\mathsf{pkt} \in \mathbf{Pkt}$, $d \in \mathbf{Data}$, and $pt \in \mathbf{Path}(t_1)$. For $g \in \mathcal{F}$ such that $g(d)$ holds true,

$$\begin{aligned}
&\text{(by Theorem 3.4)} & \mathsf{proj}_g(\overline{f}(\mathsf{push}(q_1, \mathsf{pkt}, d, pt))) &= \widehat{f}(\mathsf{proj}_g(\mathsf{push}(q_1, \mathsf{pkt}, d, pt))) \\
&\text{(by Lemma 2.5)} & &= \widehat{f}(\mathsf{push}(\mathsf{proj}_g(q_1), \mathsf{pkt}, pt)) \\
&\text{(by [MLF+23, Lemma 5.9])} & &= \mathsf{push}(\widehat{f}(\mathsf{proj}_g(q_1)), \mathsf{pkt}, \widetilde{f}(pt)) \\
&\text{(by Theorem 3.4)} & &= \mathsf{push}(\mathsf{proj}_g(q_2), \mathsf{pkt}, \widetilde{f}(pt)) \\
&\text{(by Lemma 2.5)} & &= \mathsf{proj}_g(\mathsf{push}(q_2, \mathsf{pkt}, d, \widetilde{f}(pt)))
\end{aligned}$$

For $g \in \mathcal{F}$ such that $g(d)$ does not hold true,

$$\begin{aligned}
&\text{(by Theorem 3.4)} & \mathsf{proj}_g(\overline{f}(\mathsf{push}(q_1, \mathsf{pkt}, d, pt))) &= \widehat{f}(\mathsf{proj}_g(\mathsf{push}(q_1, \mathsf{pkt}, d, pt))) \\
&\text{(by Lemma 2.5)} & &= \widehat{f}(\mathsf{proj}_g(q_1)) \\
&\text{(by Theorem 3.4)} & &= \mathsf{proj}_g(q_2) \\
&\text{(by Lemma 2.5)} & &= \mathsf{proj}_g(\mathsf{push}(q_2, \mathsf{pkt}, d, \widetilde{f}(pt)))
\end{aligned}$$

Overall, $\mathsf{proj}_g(\overline{f}(\mathsf{push}(q_1, \mathsf{pkt}, d, pt))) = \mathsf{proj}_g(\mathsf{push}(q_2, \mathsf{pkt}, d, \widetilde{f}(pt)))$ for all $g \in \mathcal{F}$. Hence,

$$\overline{f}(\mathsf{push}(q_1, \mathsf{pkt}, d, pt)) = \mathsf{push}(q_2, \mathsf{pkt}, d, \widetilde{f}(pt))$$

by Lemma 2.2, as desired.

$\square$

References

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