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. 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

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

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

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

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.

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

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 $\text{proj}_f : \textbf{PIEOTree}(t) \to \textbf{PIFOTree}(t)$ by

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

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

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

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

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

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

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

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

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

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

□

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

Proof. We'll do induction on t.

(Leaf) For $t = *$, let $q = \text{Leaf}(p)$ and $\text{proj}_f(q) = \text{Leaf}(p')$.

$$\begin{aligned} \text{pop}(q, f) \text{ is undefined} &\iff p \text{ has no items with meta-data satisfying } f \\ &\iff p' \text{ is empty} \\ &\iff \text{pop}(\text{proj}_f(q)) \text{ is undefined} \end{aligned}$$

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

$$p \text{ has no items with meta-data satisfying } f \iff p' \text{ is empty}$$

and

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

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

□

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

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

Proof. TODO □

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

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

Proof. TODO □

3 Embedding & Simulation

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

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

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

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

- For $t_1 = *$, define $\overline{f}(q) = q$. This is well-defined by [MLF+23, Lemma 5.2].
- For $t_1 = \text{Node}(ts_1)$, $n = |ts_1|$, $q = \text{Internal}(qs, p)$, construct $\overline{f}_\alpha(q) \in$ **PIEOTree**(t_2/α) 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 $\overline{f}(q)_{\alpha \cdot j} \in$ **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 $\overline{f}(q)_\alpha = \text{Internal}(qs_\alpha, p_\alpha)$.

Theorem 3.4. *The following diagram commutes*

$$\begin{array}{ccc} \textbf{PIEOTree}(t_1) & \xrightarrow{\overline{f}} & \textbf{PIEOTree}(t_2) \\ \downarrow{\scriptstyle \text{proj}_g} & & \downarrow{\scriptstyle \text{proj}_g} \\ \textbf{PIFOTree}(t_1) & \xrightarrow{\widehat{f}} & \textbf{PIFOTree}(t_2) \end{array}$$

*In other words, for $q_1 \in$ **PIEOTree**(t_1), $q_2 = \overline{f}(q_1) \in$ **PIEOTree**(t_2), and $g \in \mathcal{F}$, $\text{proj}_g(q_2) = \widehat{f}(\text{proj}_g(q_1))$.*

Proof. TODO □

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

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

is a simulation.

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

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

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

By Lemma 2.4,

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

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,

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

Since our choice of g was arbitrary, notice Equation ($\dagger$) 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,

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

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

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

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.

□

References

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