PIEO Trees for Fun and Profit

We assume familiarity with [Mohan et al., 2023], adopt its notational conventions, and borrow 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 $\mathbf{Rk}$
(3) support predicates in F
(4) admit partial functions

$$\text{pop} : \mathbf{PIEO}(S, D, F) \times F \rightharpoonup S \times \mathbf{PIEO}(S, D, F)$$
$$\text{push} : \mathbf{PIEO}(S, D, F) \times S \times D \times \mathbf{Rk} \to \mathbf{PIEO}(S, D, F)$$

For $p \in \mathbf{PIEO}(S, D, F)$, $s \in S$, and $f \in F$, 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 $f, f' \in \mathcal{F}$, $f \leq f' \land f(d) \implies f'(d)$.

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

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

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

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

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.

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

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 Embedding and Simulation

Definition 2.1. Let $t_1, t_2 \in \textbf{Topo}$. We call a relation $R \subseteq \textbf{PIEOTree}(t_1) \times \textbf{PIEOTree}(t_2)$ a *simulation* if, for all pkt $\in \textbf{Pkt}$, $f \in \mathcal{F}$, 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 $pt_1 \in \textbf{Path}(t_1)$ and $d \in \textbf{Data}$, there exists $pt_2 \in \textbf{Path}(t_2)$ 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 2.2. For all further discussion, we assume our embeddings are injective.

Definition 2.3. For $t_1, t_2 \in \textbf{Topo}$, let f be an embedding from t_1 to t_2. We lift f to a map $\overline{f}$ from $\textbf{PIEOTree}(t_1)$ to $\textbf{PIEOTree}(t_2)$ inductively.

- For $t_1 = *$, define $\overline{f}(q) = q$. This is well-defined by [Mohan et al., 2023, Lemma 5.2].
- For $t_1 = \text{Node}(ts_1)$, $n = |ts_1|$, $q = \text{Internal}(qs, p)$, construct $\overline{f}_\alpha(q) \in \textbf{PIEOTree}(t_2/\alpha)$ 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 [Mohan et al., 2023, 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 $\overline{f}(q)_{\alpha \cdot j} \in \textbf{PIEOTree}(t_2/(\alpha \cdot j))$ 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 2.4. *Let $t_1, t_2 \in \textbf{Topo}$. If f embeds t_1 into t_2, then*

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

is a simulation.

References

[Mohan et al., 2023] Mohan, A., Liu, Y., Foster, N., Kappé, T., and Kozen, D. (2023). Formal abstractions for packet scheduling.