

## 1 Joins on $Query(v)$ as a monoid

In order to justify some of our optimisations, we need to know certain properties of join denotation.

Firstly, we want to define establish closure and associativity of the *join* function defined in the denotational semantics.

We want to define these for queryable subsets of a view  $v \in View_\Sigma$

$$Query(v) = \{s \mid \exists P, A, B. s = \llbracket P \rrbracket(v) \wedge \Sigma \vdash P: A, B\} \quad (1)$$

### 1.1 Lemma: Join is closed on $Query(v)$

Proof: For  $s, t \in Query(v)$  there exists  $P, Q, A, B, C, D$  such that  $\Sigma \vdash P: A, B$  and  $\Sigma \vdash Q: C, D$  hold and  $s = \llbracket P \rrbracket(v) \wedge t = \llbracket Q \rrbracket(v)$

**Case  $B \neq C$**  Then the join is empty, since no  $b \in B = c \in C$ , so

$$join(\llbracket P \rrbracket(v), \llbracket Q \rrbracket(v)) = \emptyset = \llbracket Distinct(Id_T) \rrbracket(v) \quad (2)$$

For some type  $T \in \Sigma$  (We know  $\Sigma$  contains more than one type, due to the typing of  $P$  and  $Q$ )

**Case  $B = C$**  Then the join is not empty. By the correspondence of denotational and operational semantics, we have

$$join(\llbracket P \rrbracket(v), \llbracket Q \rrbracket(v)) = \llbracket Chain(P, Q) \rrbracket(v) \quad (3)$$

with the following typing

$$\Sigma \vdash Chain(P, Q): A, D \quad (4)$$

### 1.2 Lemma: Join is associative on $Query(v)$

Proof: For  $r, s, t \in Query(v)$  there exist queries  $P, Q, R$  and object types  $A, B, C, D, E, F$  such that  $\Sigma \vdash P: A, B$ ,  $\Sigma \vdash Q: C, D$ ,  $\Sigma \vdash R: E, F$  and hold and  $r = \llbracket P \rrbracket(v) \wedge s = \llbracket Q \rrbracket(v) \wedge t = \llbracket R \rrbracket(v)$   
We want to prove that  $join(r, join(s, t)) = join(join(r, s), t)$

**Case  $B \neq C$**  Then

$$\begin{aligned} join(r, join(s, t)) &= join(r, u) \text{ For some } u, \text{ either } u = \emptyset \text{ or } u \text{ has a left type of } C \\ &= \emptyset \text{ hence equals the empty set} \\ &= join(\emptyset, t) \\ &= join(join(r, s), t) \end{aligned} \quad (5)$$

**Case  $D \neq E$**  Then similarly

$$\begin{aligned} join(join(r, s), t) &= join(u, t) \text{ For some } u, \text{ either } u = \emptyset \text{ or } u \text{ has a right type of } D \\ &= \emptyset \text{ hence equals the empty set} \\ &= join(r, \emptyset) \\ &= join(r, join(s, t)) \end{aligned} \quad (6)$$

**Case  $B = C$  and  $D = E$**  Then

$$\begin{aligned}
(a, f) \in \text{join}(r, \text{join}(s, t)) &\Leftrightarrow (a, f) \in \text{join}(\llbracket P \rrbracket(v), \text{join}(\llbracket Q \rrbracket(v), \llbracket R \rrbracket(v))) \\
&\Leftrightarrow a, f \triangleleft_{A, F, v} \text{Chain}(P, \text{Chain}(Q, R)) \text{ by deno-oper correspondence} \\
&\Leftrightarrow \exists b, d. (a, b) \triangleleft_{A, B, v} P \wedge (b, d) \triangleleft_{B, D, v} Q \wedge (d, f) \triangleleft_{D, F, v} R \\
&\Leftrightarrow (a, f) \triangleleft_{A, F, v} \text{Chain}(\text{Chain}(P, Q), R) \\
&\Leftrightarrow (a, f) \in \llbracket \text{Chain}(\text{Chain}(P, Q), R) \rrbracket(v) \\
&\Leftrightarrow (a, f) \in \text{join}(\text{join}(\llbracket P \rrbracket(v), \llbracket Q \rrbracket(v)), \llbracket R \rrbracket(v)) \\
&\Leftrightarrow (a, f) \in \text{join}(\text{join}(r, s), t)
\end{aligned} \tag{7}$$

Hence  $\text{Query}(v)$  is a monoid with  $\text{join}$ . However, it doesn't particularly stick to a nice typescheme.

### 1.3 Joins on $\text{Query}_A(v)$ as a monoid

If we now take the more useful subset  $\text{Query}_A(v)$  of  $\text{Query}(v)$

$$\text{Query}_A(v) = \{s \subseteq A \times A \mid \exists P. \Sigma \vdash P: A, A \wedge s = \llbracket P \rrbracket(v)\} \tag{8}$$

Trivially, from the above,  $\text{join}$  over  $\text{Query}_A(v)$  also forms a monoid.

This enables several optimisations.

Firstly, writing  $p$  for  $\llbracket P \rrbracket(v)$  and with the binary representation of  $n = \sum_i b_i * 2^i$

$$\begin{aligned}
\llbracket \text{Exactly}(n, P) \rrbracket(v) &= p^n \text{ In } \text{Query}_A(v) \\
&= \prod_i p^{b_i * 2^i}
\end{aligned} \tag{9}$$