

1 Semantics

1.1 Grammar and definitions

Value expressions

$$\begin{aligned} V \rightarrow C^\gamma & \text{ Typed constants} \\ | () & \text{ Unit constructor} \\ | \lambda x : \tau. M & \text{ term lambda expressions} \\ | x & \text{ Variables} \\ | \Lambda \phi. V & \text{ Effect lambda expressions} \\ | V \epsilon & \text{ Effect specialisation} \end{aligned} \tag{1}$$

Computation expressions

$$\begin{aligned} M \rightarrow V & \text{ VApplication} \\ | do\ x \leftarrow M\ in\ M & \text{ Sequencing} \\ | return\ V & \text{ Unit computation} \\ | Cons^{\tau_1 \rightarrow T_\xi \tau_2}(V) & \text{ Computation Constructors} \\ | Op^{T_\xi \tau_1 \rightarrow T_\xi \tau_2}(M) & \text{ Computation Combinators} \end{aligned} \tag{2}$$

Value types

$$\begin{aligned} \tau \rightarrow \gamma & \text{ Ground Types} \\ | \tau \rightarrow \Theta & \text{ Functions are kleisli arrows} \\ | \forall \phi. \tau & \text{ Effect quantification} \end{aligned} \tag{3}$$

Computation (Monad) types

$$\Theta \rightarrow T_\epsilon \tau \text{ Monad type constructor} \tag{4}$$

Effects

$$\epsilon \rightarrow \phi \mid \xi.. \text{ Effect variables or ground effects}$$

1.2 typing

Value typing rules

$$\begin{aligned}
& (\text{Const}) \frac{\Gamma \text{ OK}}{\Gamma \vdash C^\gamma : \gamma} \\
& (\text{Unit}) \frac{\Gamma \text{ OK}}{\Gamma \vdash () : \text{unit}} \\
& (\text{Weaken}) \frac{\Gamma \vdash x : A}{\Gamma, x' : A' \vdash x : A} (x \neq x') \\
& (\text{Var}) \frac{\Gamma \text{ OK}}{\Gamma, x : A \vdash x : A} \\
& (\text{Eff-Gen}) \frac{\Gamma \vdash V : A}{\Gamma \vdash \Lambda \phi. V : \forall \phi. A} \phi \notin \text{fev}(\Gamma) \\
& (\text{Eff-Spec}) \frac{\Gamma \vdash V : \forall \phi. A}{\Gamma \vdash V \epsilon : A[\epsilon \phi]}
\end{aligned}$$

The rules for types of Single queries are similar:

$$\begin{aligned}
& (\text{Find}) \frac{\Sigma(f) \downarrow (A)}{\Gamma \vdash \text{Find}(f) : A} \\
& (\text{From}) \frac{\Gamma \vdash P : (A, B) \quad \Gamma \vdash S : A}{\Gamma \vdash \text{From}(S, P) : B} \\
& (\text{AndS}) \frac{\Gamma \vdash S : A \quad \Gamma \vdash S' : A}{\Gamma \vdash \text{AndS}(S, S') : A} \\
& (\text{OrS}) \frac{\Gamma \vdash S : A \quad \Gamma \vdash S' : A}{\Gamma \vdash \text{OrS}(S, S') : A}
\end{aligned}$$

1.3 Operational Semantics

Now we shall define a set of rules for determining if a pair of objects is a valid result of a query. We're interested in forming a relation $a \triangleleft p$ to mean “a is a valid result of query Q”. This is dependent on the current view $v : \text{View}_\Sigma$, and the type of the expression. Hence we define $(a, b) \triangleleft_{(A, B), v} P$ for pair queries P and $a \triangleleft_{A, v} S$ for single queries S .

$$\begin{aligned}
& (\text{Rel}) \frac{(a, b) \in v(r)}{(a, b) \triangleleft_{(A,B),v} \text{Rel}(r)} \\
& (\text{Rev}) \frac{(b, a) \in v(r)}{(a, b) \triangleleft_{(A,B),v} \text{RevRel}(r)} \\
& (\text{Id}) \frac{a \in v(A)}{(a, a) \triangleleft_{(A,A),v} \text{Id}_A} \\
& (\text{Distinct}) \frac{(a, b) \triangleleft_{(A,B),v} P \quad a \neq b}{(a, b) \triangleleft_{(A,B),v} \text{Distinct}(P)} \\
& (\text{And}) \frac{(a, b) \triangleleft_{(A,B),v} P \quad (a, b) \triangleleft_{(A,B),v} Q}{(a, b) \triangleleft_{(A,B),v} \text{And}(P, Q)} \\
& (\text{Or1}) \frac{(a, b) \triangleleft_{(A,B),v} P}{(a, b) \triangleleft_{(A,B),v} \text{Or}(P, Q)} \\
& (\text{Or2}) \frac{(a, b) \triangleleft_{(A,B),v} Q}{(a, b) \triangleleft_{(A,B),v} \text{Or}(P, Q)} \\
& (\text{Chain}) \frac{(a, b) \triangleleft_{(A,B),v} P \quad (b, c) \triangleleft_{(B,C),v} Q}{(a, c) \triangleleft_{(A,C),v} \text{Chain}(P, Q)} \\
& (\text{AndLeft}) \frac{(a, b) \triangleleft_{(A,B),v} P \quad a \triangleleft_{A,v} S}{(a, b) \triangleleft_{(A,B),v} \text{AndLeft}(P, S)} \\
& (\text{AndRight}) \frac{(a, b) \triangleleft_{(A,B),v} P \quad b \triangleleft_{B,v} S}{(a, b) \triangleleft_{(A,B),v} \text{AndRight}(P, S)} \\
& (\text{Exactly} \cdot 0) \frac{(a, b) \triangleleft_{(A,A),v} \text{Id}_A}{(a, b) \triangleleft_{(A,A),v} \text{Exactly}(0, P)} \\
& (\text{Exactly} \cdot n+1) \frac{(a, b) \triangleleft_{(A,A),v} P \quad (b, c) \triangleleft_{(A,A),v} \text{Exactly}(n, P)}{(a, c) \triangleleft_{(A,A),v} \text{Exactly}(n+1, P)} \\
& (\text{Upto} \cdot 0) \frac{(a, b) \triangleleft_{(A,A),v} \text{Id}_A}{(a, b) \triangleleft_{(A,A),v} \text{Upto}(0, P)} \\
& (\text{Upto} \cdot n) \frac{(a, b) \triangleleft_{(A,A),v} \text{Upto}(n, P)}{(a, b) \triangleleft_{(A,A),v} \text{Upto}(n+1, P)} \\
& (\text{Upto} \cdot n+1) \frac{(a, b) \triangleleft_{(A,A),v} P \quad (b, c) \triangleleft_{(A,A),v} \text{Upto}(n, P)}{(a, c) \triangleleft_{(A,A),v} \text{Upto}(n+1, P)} \\
& (\text{fix1}) \frac{(a, b) \triangleleft_{(A,A),v} \text{Id}_A}{(a, b) \triangleleft_{(A,A),v} \text{FixedPoint}(P)} \\
& (\text{fix2}) \frac{(a, b) \triangleleft_{(A,A),v} P \quad (b, c) \triangleleft_{(A,A),v} \text{FixedPoint}(P)}{(a, b) \triangleleft_{(A,A),v} \text{FixedPoint}(P)}
\end{aligned}$$

And the FindSingle rules

$$\begin{array}{c}
(\text{Find}) \frac{a \in v(A) \quad f(a) \downarrow \text{True}}{a \triangleleft_{A,v} \text{Find}(f)} \\
(\text{From}) \frac{a \triangleleft_{A,v} S \quad (a, b) \triangleleft_{(A,B),v} P}{b \triangleleft_{A,v} \text{From}(S, P)} \\
(\text{AndS}) \frac{a \triangleleft_{A,v} S \quad a \triangleleft_{A,v} S'}{a \triangleleft_{A,v} \text{And}(S, S')} \\
(\text{OrS1}) \frac{a \triangleleft_{A,v} S}{a \triangleleft_{A,v} \text{Or}(S, S')} \\
(\text{OrS1}) \frac{a \triangleleft_{A,v} S'}{a \triangleleft_{A,v} \text{Or}(S, S')}
\end{array}$$

1.4 Denotational Semantics

The operational semantics clearly demonstrate membership of a query, but don't give a means to efficiently generate the results of query. To this end, we introduce denotations $\llbracket P \rrbracket$ and $\llbracket S \rrbracket$ such that

$$\Sigma \vdash P: (A, B) \Rightarrow \llbracket P \rrbracket: \text{View}_\Sigma \rightarrow \wp(A \times B)$$

and

$$\Sigma \vdash S: A \Rightarrow \llbracket S \rrbracket: \text{View}_\Sigma \rightarrow \wp(A)$$

Such denotations should be compositional and syntax directed, whilst still corresponding to the operational semantics.

$$\llbracket Rel(r) \rrbracket(v) = v(r)$$

$$\llbracket RevRel(r) \rrbracket(v) = swap(v(r))$$

$$\llbracket Id_A \rrbracket(v) = dup(v(A))$$

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

$$\llbracket And(P, Q) \rrbracket(v) = \llbracket P \rrbracket(v) \cap \llbracket Q \rrbracket(v)$$

$$\llbracket Or(P, Q) \rrbracket(v) = \llbracket P \rrbracket(v) \cup \llbracket Q \rrbracket(v)$$

$$\llbracket AndLeft(P, S) \rrbracket(v) = filterLeft(\llbracket P \rrbracket(v), \llbracket S \rrbracket(v))$$

$$\llbracket AndRight(P, S) \rrbracket(v) = filterRight(\llbracket P \rrbracket(v), \llbracket S \rrbracket(v))$$

$$\llbracket Distinct(P) \rrbracket(v) = distinct(\llbracket P \rrbracket(v))$$

$$\llbracket Exactly(n, P) \rrbracket(v) = (\lambda pairs. join(\llbracket P \rrbracket(v), pairs))^n \llbracket Id_A \rrbracket(v)$$

$$\llbracket Upto(n, P) \rrbracket(v) = (\lambda pairs. join(\llbracket P \rrbracket(v), pairs) \cup pairs)^n \llbracket Id_A \rrbracket(v)$$

$$\llbracket FixedPoint(P) \rrbracket(v) = fix(\lambda pairs. join(\llbracket P \rrbracket(v), pairs) \cup pairs) \text{ in the domain } closure(A, v)$$

And similarly with single queries

$$\llbracket Find(f) \rrbracket(v) = \{a \in v(A) \mid f(a) \downarrow True\} \text{ for } \Sigma(f) = A$$

$$\llbracket From(S, P) \rrbracket(v) = \{b \mid (a, b) \in \llbracket P \rrbracket(v) \wedge a \in \llbracket S \rrbracket(v)\}$$

$$\llbracket AndS(S, S') \rrbracket(v) = \llbracket S \rrbracket(v) \cap \llbracket S' \rrbracket(v)$$

$$\llbracket OrS(S, S') \rrbracket(v) = \llbracket S \rrbracket(v) \cup \llbracket S' \rrbracket(v)$$

with the following definitions:

$$swap(s) = \{(b, a) \mid (a, b) \in s\}$$

$$dup(s) = \{(a, a) \mid a \in s\}$$

$$join(p, q) = \{(a, c) \mid \exists b. (a, b) \in p \wedge (b, c) \in q\}$$

$$distinct(s) = \{(a, b) \in s \mid a \neq b\}$$

$$filterLeft(p, s) = \{(a, b) \in p \mid a \in s\}$$

$$filterRight(p, s) = \{(a, b) \in p \mid b \in s\}$$

1.5 The domain closure(A, v)

For the subsequent proofs it is necessary to define the scott domain $\text{closure}(A, v)$ for some object type A and $\text{View}_\Sigma v$. This domain is the set of subsets x such that $\llbracket Id_A \rrbracket(v) \subseteq x \subseteq A \times A$ with bottom element $\perp = \llbracket Id_A \rrbracket(v)$ and partial order $x \sqsubseteq y \Leftrightarrow x \subseteq y$. From this point on, I shall use $x \subseteq y$ to mean $y \sqsubseteq x$.

Theorem: $\text{closure}(A, v)$ is a domain Firstly, by definition,

$$\forall x. x \in \text{closure}(A, v) \Rightarrow x \supseteq \llbracket Id_A \rrbracket(v)$$

hence $\llbracket Id_A \rrbracket(v)$ is the bottom element.

Secondly for any chain $x_1 \subseteq x_2 \subseteq x_3 \subseteq \dots, x_i \in \text{closure}(A, v)$, there exists a value $\bigsqcup_n x_n \in \text{closure}(A, v)$ such that $\forall i. \bigsqcup_n x_n \supseteq x_i$ and $\forall y. (\forall i. x_i \subseteq y) \Rightarrow y \supseteq \bigsqcup_n x_n$

proof: Take $\bigsqcup_n x_n = \bigcup_n x_n$. This is in $\text{closure}(A, v)$, since both $\bigcup_n x_n \supseteq \llbracket Id_A \rrbracket(v)$ due to $\forall i. x_i \supseteq \llbracket Id_A \rrbracket(v)$ by definition and $\bigcup_n x_n \subseteq A \times A$, by

$$\forall a. (a \in \bigcup_n x_n \wedge \neg(a \in A \times A)) \Rightarrow (\exists i. a \in x_i \wedge \neg a \in A \times A) \Rightarrow (\exists i. \neg x_i \subseteq A \times A)$$

yielding a contradiction if $\bigcup_n x_n \subseteq A \times A$ does not hold.

We know $\forall i. \bigcup_n x_n \supseteq x_i$ by definition, so it is an upper bound.

To prove it is a least upper bound, consider y such that $(\forall i.) y \supseteq x_i$

$$\begin{aligned} \forall a. (\exists i. a \in x_i) &\Rightarrow a \in y \\ \forall a. \bigvee_n (a \in x_n) &\Rightarrow a \in y \\ \forall a. (a \in \bigcup_n x_n) &\Rightarrow a \in y \\ \therefore \bigcup_n x_n &\subseteq y \end{aligned} \tag{5}$$

□

1.6 Correspondence of operational and denotational semantics

In order to use the denotational semantics to construct an interpreter or compiler we need to prove they are equivalent to the operational semantics. Namely:

For any pair query P , schema Σ and $\text{View}_\Sigma v$:

$$\Gamma \vdash P: (A, B) \Rightarrow (a, b \triangleleft_{(A, B), v} P \Leftrightarrow (a, b) \in \llbracket P \rrbracket(v))$$

And for any single query S , schema Σ and $\text{View}_\Sigma v$:

$$\Gamma \vdash S: A \Rightarrow (a \triangleleft_{A, v} S \Leftrightarrow a \in \llbracket S \rrbracket(v))$$

In order to prove these two propositions, we define two induction hypotheses

$$\Phi(\Sigma, P, A, B) \Leftrightarrow (\Gamma \vdash P: (A, B) \Rightarrow \forall v \in \text{View}_\Sigma, (a, b) \in A \times B. ((a, b) \triangleleft_{(A, B), v} P \Leftrightarrow (a, b) \in \llbracket P \rrbracket(v))))$$

$$\Psi(\Sigma, S, A) \Leftrightarrow (\Gamma \vdash S: A \Rightarrow \forall v \in View_\Sigma, a \in A. (a \triangleleft_{A,v} S) \Leftrightarrow (a \in \llbracket S \rrbracket(v)))$$

Now we shall induct over the structures of P and S starting with the SingleQuery cases

Case $S = AndS(S', S'')$

$$\begin{aligned} a \in \llbracket S \rrbracket(v) &\Leftrightarrow a \in (\llbracket S' \rrbracket(v) \cap \llbracket S'' \rrbracket(v)) \\ &\Leftrightarrow (a \in \llbracket S' \rrbracket(v) \wedge a \in \llbracket S'' \rrbracket(v)) \\ &\Leftrightarrow (a \triangleleft_{A,v} S' \wedge a \triangleleft_{A,v} S'') \text{ by } \Psi(\Sigma, S', A), \Psi(\Sigma, S'', A) \\ &\Leftrightarrow (a \triangleleft_{A,v} AndS(S', S'')) \text{ by inversion of (AndS)} \end{aligned} \tag{6}$$

Case $S = OrS(S', S'')$

$$\begin{aligned} a \in \llbracket S \rrbracket(v) &\Leftrightarrow a \in (\llbracket S' \rrbracket(v) \cup \llbracket S'' \rrbracket(v)) \\ &\Leftrightarrow (a \in \llbracket S' \rrbracket(v) \vee a \in \llbracket S'' \rrbracket(v)) \\ &\Leftrightarrow (a \triangleleft_{A,v} S' \vee a \triangleleft_{A,v} S'') \text{ by } \Psi(\Sigma, S', A), \Psi(\Sigma, S'', A) \\ &\Leftrightarrow (a \triangleleft_{A,v} OrS(S', S'')) \text{ by inversion of (OrS)} \end{aligned} \tag{7}$$

Case $S = From(S', P)$

$$\begin{aligned} b \in \llbracket S \rrbracket(v) &\Leftrightarrow \exists a \in A. \quad (a \in \llbracket S' \rrbracket(v) \wedge (a, b) \in \llbracket P \rrbracket(v)) \\ &\Leftrightarrow \exists a \in A. \quad (a \triangleleft_{A,v} S' \wedge (a, b) \triangleleft_{(A,B),v} P) \text{ by } \Psi(\Sigma, S', A), \Phi(\Sigma, P, A, B) \\ &\Leftrightarrow b \triangleleft_{B,v} From(S', P) \quad \text{by inversion of (OrS)} \end{aligned} \tag{8}$$

Case $S = Find(f)$

$$\begin{aligned} a \in \llbracket S \rrbracket(v) &\Leftrightarrow a \in v(A) \wedge f(a) \downarrow True \quad \text{by inversion of the type rule (Find)} \\ &\Leftrightarrow a \triangleleft_{A,v} Find(f) \text{ by definition} \end{aligned} \tag{9}$$

Now, looking at the FindPair queries

Case $P = Rel(r)$

$$\begin{aligned} (a, b) \in \llbracket P \rrbracket(v) &\Leftrightarrow (a, b) \in v(r) \\ &\Leftrightarrow (a, b) \triangleleft_{(A,B),v} Rel(r) \text{ by definition} \end{aligned} \tag{10}$$

Case $P = RevRel(r)$

$$\begin{aligned} (a, b) \in \llbracket P \rrbracket(v) &\Leftrightarrow (b, a) \in v(r) \\ &\Leftrightarrow (a, b) \triangleleft_{(A,B),v} RevRel(r) \text{ by definition} \end{aligned} \tag{11}$$

Case $P = Id_A$

$$\begin{aligned} (a, b) \in \llbracket P \rrbracket(v) &\Leftrightarrow a \in v(A) \wedge a = b \\ &\Leftrightarrow (a, b) \triangleleft_{(A,B),v} Id_A \text{ by definition} \end{aligned} \tag{12}$$

Case $P = \text{Chain}(P', Q)$

We have by inversion of the (Chain) type rule $\Gamma \vdash P : (A, C) \Leftrightarrow \exists B. \quad \Gamma \vdash P' : (A, B) \wedge \Gamma \vdash Q : (B, C)$

$$\begin{aligned}
(a, c) \in \llbracket P \rrbracket(v) &\Leftrightarrow (a, c) \in \text{join}(\llbracket P' \rrbracket(v), \llbracket Q \rrbracket(v)) \\
&\Leftrightarrow \exists b \in B. \quad (a, b) \in \llbracket P' \rrbracket(v) \wedge (b, c) \in \llbracket Q \rrbracket(v) \\
&\Leftrightarrow \exists b \in B. \quad (a, b) \triangleleft_{(A, B), v} P' \wedge (b, c) \triangleleft_{(B, C), v} Q \quad \text{by } \Phi(\Sigma, P', A, B), \Phi(\Sigma, Q, B, C) \\
&\Leftrightarrow (a, c) \triangleleft_{(A, C), v} \text{Chain}(P', Q) \text{ by definition}
\end{aligned} \tag{13}$$

Case $P = \text{And}(P', Q)$

$$\begin{aligned}
(a, b) \in \llbracket P \rrbracket(v) &\Leftrightarrow (a, b) \in (\llbracket P' \rrbracket(v) \cap \llbracket Q \rrbracket(v)) \\
&\Leftrightarrow (a, b) \in \llbracket P' \rrbracket(v) \wedge (a, b) \in \llbracket Q \rrbracket(v) \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} P' \wedge (a, b) \triangleleft_{(A, B), v} Q \quad \text{by } \Phi(\Sigma, P', A, B), \Phi(\Sigma, Q, A, B) \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} \text{And}(P', Q) \text{ by inversion of (And)}
\end{aligned} \tag{14}$$

Case $P = \text{Or}(P', Q)$

$$\begin{aligned}
(a, b) \in \llbracket P \rrbracket(v) &\Leftrightarrow (a, b) \in (\llbracket P' \rrbracket(v) \cup \llbracket Q \rrbracket(v)) \\
&\Leftrightarrow (a, b) \in \llbracket P' \rrbracket(v) \vee (a, b) \in \llbracket Q \rrbracket(v) \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} P' \vee (a, b) \triangleleft_{(A, B), v} Q \quad \text{by } \Phi(\Sigma, P', A, B), \Phi(\Sigma, Q, A, B) \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} \text{Or}(P', Q) \text{ by inversion of (Or1), (Or2)}
\end{aligned} \tag{15}$$

Case $P = \text{AndLeft}(P', S)$

$$\begin{aligned}
(a, b) \in \llbracket P \rrbracket(v) &\Leftrightarrow (a, b) \in \llbracket P' \rrbracket(v) \wedge a \in \llbracket S \rrbracket(v) \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} P' \wedge a \triangleleft_{A, v} S \quad \text{by } \Phi(\Sigma, P', A, B), \Psi(\Sigma, S, A) \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} \text{AndLeft}(P', S) \text{ by inversion of (AndLeft)}
\end{aligned} \tag{16}$$

Case $P = \text{AndRight}(P', S)$

$$\begin{aligned}
(a, b) \in \llbracket P \rrbracket(v) &\Leftrightarrow (a, b) \in \llbracket P' \rrbracket(v) \wedge b \in \llbracket S \rrbracket(v) \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} P' \wedge b \triangleleft_{B, v} S \quad \text{by } \Phi(\Sigma, P', A, B), \Psi(\Sigma, S, B) \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} \text{AndRight}(P', S) \text{ by inversion of (AndRight)}
\end{aligned} \tag{17}$$

Case $P = \text{Distinct}(P')$

$$\begin{aligned}
(a, b) \in \llbracket P \rrbracket(v) &\Leftrightarrow (a, b) \in \llbracket P' \rrbracket(v) \wedge a \neq b \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} P' \wedge a \neq b \quad \text{by } \Phi(\Sigma, P', A, B) \\
&\Leftrightarrow (a, b) \triangleleft_{(A, B), v} \text{Distinct}(P') \text{ by inversion of (AndRight)}
\end{aligned} \tag{18}$$

Case $P = \text{Exactly}(n, P')$

We have by inversion of the (Exactly) type rule $\Gamma \vdash P : (A, A) \wedge \Gamma \vdash P' : (A, A)$

$$\text{let } f = (\lambda \text{pairs}. \text{join}(\llbracket P' \rrbracket(v), \text{pairs})) \quad (19)$$

then

$$(a, b) \in \llbracket P \rrbracket(v) \Leftrightarrow (a, b) \in f^n \llbracket Id_A \rrbracket(v) \quad (20)$$

hence it suffices to prove

$$(a, b) \in f^n \llbracket Id_A \rrbracket(v) \Leftrightarrow (a, b) \triangleleft_{(A, A), v} \text{Exactly}(n, P') \quad (21)$$

Case $\text{Exactly}(0, P') :$

$$f^0 \llbracket Id_A \rrbracket(v) = \llbracket Id_A \rrbracket(v)$$

so by $\Phi(\Sigma, (a, b), A, A), Id_A$

$$\begin{aligned} (a, b) \in f^0 \llbracket Id_A \rrbracket(v) &\Leftrightarrow (a, b) \in \llbracket Id_A \rrbracket(v) \\ &\Leftrightarrow (a, b) \triangleleft_{(A, A), v} Id_A \\ &\Leftrightarrow (a, b) \triangleleft_{(A, A), v} \text{Exactly}(0, P') \end{aligned} \quad (22)$$

Case $\text{Exactly}(n+1, P')$, **assuming** $\Phi(\Sigma, \text{Exactly}(n, P'), A, A) :$

$$f^{n+1} \llbracket Id_A \rrbracket(v) = f(f^n(\llbracket Id_A \rrbracket(v)))$$

so by $\Phi(\Sigma, \text{Exactly}(n, P'), A, A)$

$$\begin{aligned} (a, b) \in f^{n+1} \llbracket Id_A \rrbracket(v) &\Leftrightarrow \exists a'. (a, a') \in \llbracket P' \rrbracket(v) \wedge (a', b) \in f^n(\llbracket Id_A \rrbracket(v)) \quad \text{by definition of } \text{join} \text{ and } f \\ &\Leftrightarrow (a, a') \triangleleft_{(A, A), v} P \wedge (a', b) \triangleleft_{(A, A), v} \text{Exactly}(n, P') \\ &\Leftrightarrow (a, b) \triangleleft_{(A, A), v} \text{Exactly}(n+1, P') \text{ by (Exactly } n+1) \end{aligned} \quad (23)$$

Case $P = \text{Upto}(n, P')$

We have by inversion of the (Upto) type rule $\Gamma \vdash P : (A, A) \wedge \Gamma \vdash P' : (A, A)$

$$\text{let } f = (\lambda \text{pairs}. \text{join}(\llbracket P' \rrbracket(v), \text{pairs}) \cup \text{pairs}) \quad (24)$$

then

$$\llbracket P \rrbracket(v) = f^n \llbracket Id_A \rrbracket(v) \text{ We now case split on } n \quad (25)$$

Case $\text{Upto}(0, P')$

$$\begin{aligned} (a, b) \in \llbracket \text{Upto}(0, P') \rrbracket(v) &\Leftrightarrow (a, b) \in f^0 \llbracket Id_A \rrbracket(v) \\ &\Leftrightarrow (a, b) \in \llbracket Id_A \rrbracket(v) \\ &\Leftrightarrow (a, b) \triangleleft_{(A, A), v} Id_A \text{ by } \Phi(\Sigma, Id_A, A, A) \\ &\Leftrightarrow (a, b) \triangleleft_{(A, A), v} \text{Upto}(0, P') \text{ by (Upto0)} \end{aligned} \quad (26)$$

Case $Upto(n + 1, P')$

$$\begin{aligned}
(a, b) \in \llbracket Upto(m + 1, P') \rrbracket(v) &\Leftrightarrow (a, b) \in f^{m+1} \llbracket Id_A \rrbracket(v) \\
&\Leftrightarrow (a, b) \in (join(\llbracket P' \rrbracket(v), f^m \llbracket Id_A \rrbracket(v)) \cup f^m \llbracket Id_A \rrbracket(v)) \\
&\Leftrightarrow (a, b) \in join(\llbracket P' \rrbracket(v), \llbracket Upto(m, P) \rrbracket(v)) \vee (a, b) \in \llbracket Upto(m, P') \rrbracket(v) \\
&\Leftrightarrow (\exists a'. (a, a') \in \llbracket P' \rrbracket(v) \wedge (a', b) \in \llbracket Upto(m, P') \rrbracket(v)) \vee (a, b) \triangleleft_{(A, A), v} Upto(m, P') \\
&\Leftrightarrow (\exists a'. (a, a') \triangleleft_{(A, A), v} P' \wedge (a', b) \triangleleft_{(A, A), v} Upto(m, P')) \vee (a, b) \triangleleft_{(A, A), v} Upto(m, P') \\
&\Leftrightarrow (a, b) \triangleleft_{(A, A), v} Upto(m + 1, P') \text{ by } (Upto\ n+1), (Upto\ n)
\end{aligned} \tag{27}$$

case $P = FixedPoint(P')$

We have by inversion of the (FixedPoint) type rule $\Gamma \vdash P : (A, A) \wedge \Gamma \vdash P' : (A, A)$

$$\begin{aligned}
&\text{let } f = (\lambda pairs. join(\llbracket P' \rrbracket(v), pairs) \cup pairs) \\
&\quad \text{then} \\
&\llbracket P \rrbracket(v) = fix(f) \text{ In the domain } closure(A, v)
\end{aligned} \tag{28}$$

Lemma: f is continuous in the domain $closure(A, v)$

Firstly, f is monotonous.

Let $x \subseteq y$

$$\begin{aligned}
(a, b) \in f(x) &\Rightarrow (a, b) \in x \vee (\exists a'. (a, a') \in \llbracket P' \rrbracket(v) \wedge (a', b) \in x) \\
&\Rightarrow (a, b) \in y \vee (\exists a'. (a, a') \in \llbracket P' \rrbracket(v) \wedge (a', b) \in y) \\
&\Rightarrow (a, b) \in f(y) \\
\therefore f(x) &\subseteq f(y)
\end{aligned} \tag{29}$$

Secondly, f preserves the lubs of chains.

Consider a chain $x_1 \subseteq x_2 \subseteq \dots$ in $closure(A, v)$ Since $closure(A, v)$ is a domain, the *lub*, $\bigcup_n x_n$ is also in $closure(A, v)$

$$\begin{aligned}
\forall m. x_m &\subseteq \bigcup_n x_n \\
\forall m. f(x_m) &\subseteq f(\bigcup_n x_n) \\
\therefore \bigcup_n f(x_n) &\subseteq f(\bigcup_n x_n)
\end{aligned} \tag{30}$$

To get the inverse relation,

$$(a, b) \in f(\bigcup_n x_n) \Rightarrow ((\exists n. (a, b) \in x_n) \vee (\exists m, a'. (a, a') \in \llbracket P' \rrbracket(v) \wedge (a', b) \in x_m)) \tag{31}$$

let $n' = \max(n, m)$ so $x_n \subseteq x_{n'} \wedge x_m \subseteq x_{n'}$

$$\begin{aligned}
& \exists n'. ((a, b) \in x_{n'}) \vee (\exists a'. (a, a') \in \llbracket P' \rrbracket(v) \wedge (a', b) \in x_{n'}) \\
& \therefore \exists n'. (a, b) \in f(x_{n'}) \\
& \therefore \exists n'. f(\bigcup_n x_n) \subseteq f(x_{n'}) \\
& \therefore f(\bigcup_n x_n) \subseteq \bigcup_n f(x_{n'})
\end{aligned} \tag{32}$$

So f is Scott-continuous.

Now, by Tarski's fixed point theorem

$$\llbracket \text{FixedPoint}(P') \rrbracket(v) = \text{fix}(f) = \bigsqcup_n f^n(\perp)$$

Lemma $(a, b) \triangleleft_{(A,A),v} \text{FixedPoint}(P') \Leftrightarrow \exists n. (a, b) \in f^n(\perp)$ Firstly, in the forwards direction, $(a, b) \triangleleft_{(A,A),v} \text{FixedPoint}(P') \Rightarrow \exists n. (a, b) \in f^n(\perp)$

By inversion of the operational rules (FixedPoint0), (FixedPoint n) and $(a, b) \triangleleft_{(A,A),v} \text{FixedPoint}(P')$, we get two cases.

Case $(a, b) \triangleleft_{A,A,v} \text{Id}_A$:

by $\Phi(\Sigma, \text{Id}_A, A, A)$, $(a, b) \in \llbracket \text{Id}_A \rrbracket(v) = \perp$
so $n = 0$

Case $(a, b) \triangleleft_{(A,A),v} P' \wedge (b, c) \triangleleft_{(A,A),v} \text{FixedPoint}(P')$

(hence $(a, c) \triangleleft_{(A,A),v} \text{FixedPoint}(P')$)

by $\Phi(\Sigma, P', A, A)$, and $(b, c) \triangleleft_{(A,A),v} \text{FixedPoint}(P')$

$$(a, b) \in \llbracket P' \rrbracket(v) \wedge \exists n. (b, c) \in f^n(\perp)$$

Instantiating with $m = n$ gives

$$(a, b) \in \llbracket P' \rrbracket(v) \wedge (b, c) \in f^m(\perp)$$

so $(a, c) \in f(f^m(\perp)) = f^{m+1}(\perp)$

Hence $(a, b) \triangleleft_{(A,A),v} \text{FixedPoint}(P') \Rightarrow \exists n. (a, b) \in f^n(\perp)$

To go the other way, we need to prove $(a, b) \in \bigsqcup_n f^n(\perp) \Rightarrow (a, b) \triangleleft_{(A,A),v} \text{FixedPoint}(P')$

$(a, b) \in \bigsqcup_n f^n(\perp)$ Means that either:

Case $(a, b) \in \perp$

$$\begin{aligned}
& \therefore (a, b) \in \llbracket \text{Id}_A \rrbracket(v) \\
& \therefore (a, b) \triangleleft_{(A,A),v} \text{Id}_A \text{ By } \Phi(\Sigma, \text{Id}_A, A, A) \\
& \therefore (a, b) \in \llbracket \text{Id}_A \rrbracket(v) \text{ By (Fix1)}
\end{aligned} \tag{33}$$

Case $\exists n \geq 0. (a, b) \in f^{n+1}(\perp) \wedge \neg((a, b) \in f^n(\perp))$

$$\begin{aligned}
& (\exists a'. (a, a') \in \llbracket P' \rrbracket(v) \wedge (a', b) \in f^n(\perp)) \vee (a, b) \in f^n(\perp) \wedge \neg((a, b) \in f^n(\perp)) \\
& \quad \therefore \exists a'. (a, a') \in \llbracket P' \rrbracket(v) \wedge (a', b) \in f^n(\perp) \\
& \quad \therefore (a, a') \triangleleft_{(A,A),v} P' \wedge (a', b) \triangleleft_{(A,A),v} \text{FixedPoint}(P') \\
& \quad \therefore (a, b) \triangleleft_{(A,A),v} \text{FixedPoint}(P')
\end{aligned} \tag{34}$$

so we have $(a, b) \triangleleft_{(A,A),v} \text{FixedPoint}(P') \Leftrightarrow \exists n. (a, b) \in f^n(\perp) \Leftrightarrow (a, b) \in \llbracket \text{FixedPoin}(P') \rrbracket(v)$

□

1.7 Write Semantics

We have fairly simple write semantics, we define the type of the *write* function as mapping a view and a set of pairs related by a relation to a new view.

$$\text{write}: \text{View}_\Sigma \rightarrow \wp(A \times R \times B) \rightarrow \text{View}_\Sigma \text{ For } A, B \in \tau \tag{35}$$

We define *write* as so:

let $rs = \{(a_i, r_i, b_i) \in (A \times R \times B) \mid 0 < i \leq n\}$ for some n being the size of the set.

$$\begin{aligned}
\text{write}(v)(rs) = & v[A \mapsto v_{\text{table}}(A) \cup \{a_i \mid 0 < i \leq n\}] \\
& [B \mapsto v_{\text{table}}(B) \cup \{b_i \mid 0 < i \leq n\}] \\
& [r \mapsto v_{\text{rel}}(R) \cup \{(a_i, b_i) \mid 0 < i \leq n\}]
\end{aligned} \tag{36}$$