1 Semantics

1.1 Grammar and definitions

FindPair queries

$$
\begin{aligned}
P \to\ & Rel(R) \text{ Find pairs related by the relation R} \\
& \mid RevRel(R) \text{ Find pairs related by the relation R in the reverse direction} \\
& \mid Chain(P, P) \text{ Find pairs related by the first subquery followed by the second} \\
& \mid And(P, P) \text{ Find pairs related by both of the sub-queries} \\
& \mid AndRight(P, S) \text{ Find pairs related by P where the right value is a result of s} \\
& \mid AndLeft(P, S) \text{ Find pairs related by P where the left value is a result of s} \\
& \mid Or(P, P) \text{ Find pairs related by either of the sub-queries} \\
& \mid Distinct(P) \text{ Find pairs related by P that are not symmetrical} \\
& \mid Id_A \text{ Identity relation} \\
& \mid Exactly(n, P) \text{ Find pairs related by n repetitions of P} \\
& \mid Upto(n, P) \text{ Find pairs related by upto n repetitions of P} \\
& \mid FixedPoint(P) \text{ Find the transitive closure of P}
\end{aligned}
\tag{1}
$$

where n denotes a natural number. These queries lookup a set of pairs of obejcts.

FindSingle queries

$$
\begin{aligned}
S \to\ & Find(F) \text{ Find values that match the findable F} \\
& \mid From(S, P) \text{ Find values that are reable from results of S via P} \\
& \mid AndS(S, S') \text{ Find values that are results of both subqueries} \\
& \mid OrS(S, S') \text{ Find values that are results of either subquery}
\end{aligned}
\tag{2}
$$

Which lookup sets of single objects.

Object Types

$$\tau \to A \mid B \mid C \mid ..$$

Which are the "real world" types stored in the database. These correspond to the user's scala classes.

Named relations

$$R \to r_1 \mid r_2 \mid ..$$

Findables

$$F_A \to f_1 \mid f_2 \mid ...$$

which are names for defined partial functions

$$f \colon A \rightharpoonup \{True, False\}$$

For some given object type A. (ie a findable is an index)

A schema Σ is made up of three partial functions:

$$\Sigma_{rel}: R \rightharpoonup \tau \times \tau$$

$$\Sigma_{findable}: F \rightharpoonup \tau$$

$$\Sigma_{table}: \tau \rightharpoonup \{True, False\}$$

Which, give the types of relations and findables, and validiate the existence of a type. When it is obvious from the context, I shall use simply use $\Sigma(x)$ to signify application of the appropriate function.

A view $v \in V_\Sigma$, for a given schema represents an immutable state of a database.

It represents a pair of partial functions. Firstly the named-relation lookup function

$$v \in V_\Sigma \Rightarrow v_{rel}(r) \in \wp(A \times B) \text{ if } \Sigma(r) \downarrow (A, B) \tag{3}$$

That is, if a relation r is in the schema, then $v(r)$ is a set of pairs of objects with object type $\Sigma(r)$. Here, and from this point onwards I am using $\wp(s)$ to represent the powerset of a set, and $f(x) \downarrow y$ to mean f is defined at x and $f(x) = y$

The next function of a view is the type-lookup function, it looks up a findable and returns

$$v \in V_\Sigma \Rightarrow v_{table}(A) \in \wp(A) \qquad \text{if } \Sigma(A) \downarrow True \tag{4}$$

That is, $v(A)$ is a set of objects of type A stored in the view, and A is a member of the schema Σ. Again I shall overload these two functions where it is clear from the context which is to be used.

1.2 typing

Typing rules take two forms. Firstly typing of pair queries:

$$\Sigma \vdash P: (A, B)$$

Which means "under the schema Σ, pair query P returns a subset of $A \times B$". The second is for single queries:

$$\Sigma \vdash S: A$$

Which means "under the schema Σ single query returns a subset of A"

The rules of the first kind are as follows

$$\text{(Rel)}\frac{\Sigma(r) \downarrow (A, B)}{\Sigma \vdash Rel(r) : (A, B)}$$

$$\text{(Rev)}\frac{\Sigma(r) \downarrow (B, A)}{\Sigma \vdash Rel(r) : (A, B)}$$

$$\text{(Id)}\frac{\Sigma(A) \downarrow True}{\Sigma \vdash Id_A : (A, A)}$$

$$\text{(Chain)}\frac{\Sigma \vdash P : (A, B) \qquad \Sigma \vdash Q : (B, C)}{\Sigma \vdash Chain(P, Q) : (A, C)}$$

$$\text{(And)}\frac{\Sigma \vdash P : (A, B) \qquad \Sigma \vdash Q : (A, B)}{\Sigma \vdash And(P, Q) : (A, B)}$$

$$\text{(Or)}\frac{\Sigma \vdash P : (A, B) \qquad \Sigma \vdash Q : (A, B)}{\Sigma \vdash Or(P, Q) : (A, B)}$$

$$\text{(Distinct)}\frac{\Sigma \vdash P : (A, B)}{\Sigma \vdash Distinct(P) : (A, B)}$$

$$\text{(AndLeft)}\frac{\Sigma \vdash P : (A, B) \qquad \Sigma \vdash S : (A)}{\Sigma \vdash AndLeft(P, S) : (A, B)}$$

$$\text{(AndRight)}\frac{\Sigma \vdash P : (A, B) \qquad \Sigma \vdash S : B}{\Sigma \vdash \Sigma \vdash AndRight(P, S) : (A, B)}$$

$$\text{(Exactly)}\frac{\Sigma \vdash P : (A, A)}{\Sigma \vdash Exactly(n, P) : (A, A)}$$

$$\text{(Upto)}\frac{\Sigma \vdash P : (A, A)}{\Sigma \vdash Upto(n, P) : (A, A)}$$

$$\text{(FixedPoint)}\frac{\Sigma \vdash P : (A, A)}{\Sigma \vdash FixedPoint(P) : (A, A)}$$

The rules for types of Single queries are similar:

$$\text{(Find)}\frac{\Sigma(f) \downarrow (A)}{\Sigma \vdash Find(f) : A}$$

$$\text{(From)}\frac{\Sigma \vdash P : (A, B) \qquad \Sigma \vdash S : A}{\Sigma \vdash From(S, P) : B}$$

$$\text{(AndS)}\frac{\Sigma \vdash S : A \qquad \Sigma \vdash S' : A}{\Sigma \vdash AndS(S, S') : A}$$

$$\text{(OrS)}\frac{\Sigma \vdash S : A \qquad \Sigma \vdash S' : A}{\Sigma \vdash OrS(S, S') : A}$$

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 $a \lhd p$ to mean"a is a valid result of query Q". This is dependent on the current view $v : View_\Sigma$, and the type of the expression. Hence we define $(a, b) \lhd_{(A,B),v} P$ for pair queries P and $a \lhd_{A,v} S$ for single queries S.

$$\text{(Rel)}\frac{(a,b) \in v(r)}{(a,b) \lhd_{(A,B),v} Rel(r)}$$

$$\text{(Rev)}\frac{(b,a) \in v(r)}{(a,b) \lhd_{(A,B),v} RevRel(r)}$$

$$\text{(Id)}\frac{a \in v(A)}{(a,a) \lhd_{(A,A),v} Id_A}$$

$$\text{(Distinct)}\frac{(a,b) \lhd_{(A,B),v} P \qquad a \neq b}{(a,b) \lhd_{(A,B),v} Distinct(P)}$$

$$\text{(And)}\frac{(a,b) \lhd_{(A,B),v} P \qquad (a,b) \lhd_{(A,B),v} Q}{(a,b) \lhd_{(A,B),v} And(P,Q)}$$

$$\text{(Or1)}\frac{(a,b) \lhd_{(A,B),v} P}{(a,b) \lhd_{(A,B),v} Or(P,Q)}$$

$$\text{(Or2)}\frac{(a,b) \lhd_{(A,B),v} Q}{(a,b) \lhd_{(A,B),v} Or(P,Q)}$$

$$\text{(Chain)}\frac{(a,b) \lhd_{(A,B),v} P \qquad (b,c) \lhd_{(B,C),v} Q}{(a,c) \lhd_{(A,C),v} Chain(P,Q)}$$

$$\text{(AndLeft)}\frac{(a,b) \lhd_{(A,B),v} P \qquad a \lhd_{A,v} S}{(a,b) \lhd_{(A,B),v} AndLeft(P,S)}$$

$$\text{(AndRight)}\frac{(a,b) \lhd_{(A,B),v} P \qquad b \lhd_{B,v} S}{(a,b) \lhd_{(A,B),v} AndRight(P,S)}$$

$$\text{(Exactly˙0)}\frac{(a,b) \lhd_{(A,A),v} Id_A}{(a,b) \lhd_{(A,A),v} Exactly(0,P)}$$

$$\text{(Exactly˙n+1)}\frac{(a,b) \lhd_{(A,A),v} P \qquad (b,c) \lhd_{(A,A),v} Exactly(n,P)}{(a,c) \lhd_{(A,A),v} Exactly(n+1,P)}$$

$$\text{(Upto˙0)}\frac{(a,b) \lhd_{(A,A),v} Id_A}{(a,b) \lhd_{(A,A),v} Upto(0,P)}$$

$$\text{(Upto˙n)}\frac{(a,b) \lhd_{(A,A),v} Upto(n,P)}{(a,b) \lhd_{(A,A),v} Upto(n+1,P)}$$

$$\text{(Upto˙n+1)}\frac{(a,b) \lhd_{(A,A),v} P \qquad (b,c) \lhd_{(A,A),v} Upto(n,P)}{(a,c) \lhd_{(A,A),v} Upto(n+1,P)}$$

$$\text{(fix1)}\frac{(a,b) \lhd_{(A,A),v} Id_A}{(a,b) \lhd_{(A,A),v} FixedPoint(P)}$$

$$\text{(fix2)}\frac{(a,b) \lhd_{(A,A),v} P \qquad (b,c) \lhd_{(A,A),v} FixedPoint(P)}{(a,b) \lhd_{(A,A),v} FixedPoint(P)}$$

And the FindSingle rules

$$\text{(Find)}\frac{a \in v(A) \qquad f(a) \downarrow True}{a \lhd_{A,v} Find(f)}$$

$$\text{(From)}\frac{a \lhd_{A,v} S \qquad (a,b) \lhd_{)(A,B),v} P}{b \lhd_{A,v} From(S,P)}$$

$$\text{(AndS)}\frac{a \lhd_{A,v} S \qquad a \lhd_{A,v} S'}{a \lhd_{A,v} And(S,S')}$$

$$\text{(OrS1)}\frac{a \lhd_{A,v} S}{a \lhd_{A,v} Or(S,S')}$$

$$\text{(OrS1)}\frac{a \lhd_{A,v} S'}{a \lhd_{A,v} Or(S,S')}$$

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 $[\![P]\!]$ and $[\![S]\!]$ such that

$$\Sigma \vdash P\colon (A,B) \Rightarrow [\![P]\!]\colon View_\Sigma \to \wp(A \times B)$$

and

$$\Sigma \vdash S\colon A \Rightarrow [\![S]\!]\colon View_\Sigma \to \wp(A)$$

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

$$[\![Rel(r)]\!](v) = v(r)$$

$$[\![RevRel(r)]\!](v) = swap(v(r))$$

$$[\![Id_A]\!](v) = dup(v(a))$$

$$[\![Chain(P,Q)]\!](v) = join([\![P]\!](v), [\![Q]\!](v))$$

$$[\![And(P,Q)]\!](v) = [\![P]\!](v) \cap [\![Q]\!](v)$$

$$[\![Or(P,Q)]\!](v) = [\![P]\!](v) \cup [\![Q]\!](v)$$

$$[\![AndLeft(P,S)]\!](v) = filterLeft([\![P]\!](v), [\![S]\!](v))$$

$$[\![AndRight(P,S)]\!](v) = filterRight([\![P]\!](v), [\![S]\!](v))$$

$$[\![Distinct(P)]\!](v) = distinct([\![P]\!](v))$$

$$[\![Exactly(n,P)]\!](v) = (\lambda pairs.join([\![P]\!](v), pairs))^n [\![Id_A]\!](v)$$

$$[\![Upto(n,P)]\!](v) = (\lambda pairs.join([\![P]\!](v), pairs) \cup pairs)^n [\![Id_A]\!](v)$$

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

And similarly with single queries

$$[\![Find(f)]\!](v) = \{a \in v(A) \mid f(a) \downarrow True\} \text{ for } \Sigma(f) = A$$

$$[\![From(S,P)]\!](v) = \{b \mid (a,b) \in [\![P]\!](v) \wedge a \in [\![S]\!](v)\}$$

$$[\![AndS(S,S')]\!](v) = [\![S]\!](v) \cap [\![S']\!](v)$$

$$[\![OrS(S,S')]\!](v) = [\![S]\!](v) \cup [\![S']\!](v)$$

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 $closure(A, v)$ for some object type A and $View_\Sigma\ v$. This domain is the set of subsets x such that $[\![Id_A]\!](v) \subseteq x \subseteq A \times A$ with bottom element $\bot = [\![Id_A]\!](v)$ 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: $closure(A, v)$ **is a domain** Firstly, by definition,

$$\forall x.x \in closure(A, v) \Rightarrow x \supseteq [\![Id_A]\!](v)$$

hence $[\![Id_A]\!](v)$ is the bottom element.
Secondly for any chain $x_1 \subseteq x_2 \subseteq x_3 \subseteq ...$, $x_i \in closure(A, v)$, there exists a value $\bigsqcup_n x_n \in closure(A, v)$ 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 $closure(A, v)$, since both $\bigcup_n x_n \supseteq [\![Id_A]\!](v)$ due to $\forall i.x_i \supseteq [\![Id_A]\!](v)$ 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 $View_\Sigma\ v$:

$$\Sigma \vdash P\colon (A, B) \Rightarrow (a, b \lhd_{(A,B),v} P \Leftrightarrow (a, b) \in [\![P]\!](v))$$

And for any single query S, schema Σ and $View_\Sigma\ v$:

$$\Sigma \vdash S\colon A \Rightarrow (a \lhd_{A,v} S \Leftrightarrow a \in [\![S]\!](v))$$

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

$$\Phi(\Sigma, P, A, B) \Leftrightarrow (\Sigma \vdash P\colon (A, B) \Rightarrow \forall v \in View_\Sigma, (a, b) \in A \times B.((a, b) \lhd_{(A,B),v} P) \Leftrightarrow (a, b) \in [\![P]\!](v))))$$

$$\Psi(\Sigma, S, A) \Leftrightarrow (\Sigma \vdash S\colon A \Rightarrow \forall v \in View_\Sigma, a \in A.(a \lhd_{A,v} S) \Leftrightarrow (a \in [\![S]\!](v)))$$

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

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

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

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

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

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

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

Case $S = Find(f)$

$$\begin{aligned} a \in [\![S]\!](v) &\Leftrightarrow a \in v(A) \wedge f(a) \downarrow True \qquad \text{by inversion of the type rule (Find)} \\ &\Leftrightarrow a \lhd_{A,v} Find(f) \text{ by definition} \end{aligned} \tag{9}$$

Now, looking at the FindPair queries

Case $P = Rel(r)$

$$\begin{aligned} (a, b) \in [\![P]\!](v) &\Leftrightarrow (a, b) \in v(r) \\ &\Leftrightarrow (a, b) \lhd_{(A,B),v} Rel(r) \text{ by definition} \end{aligned} \tag{10}$$

Case $P = RevRel(r)$

$$\begin{aligned} (a, b) \in [\![P]\!](v) &\Leftrightarrow (b, a) \in v(r) \\ &\Leftrightarrow (a, b) \lhd_{(A,B),v} RevRel(r) \text{ by definition} \end{aligned} \tag{11}$$

Case $P = Id_A$

$$\begin{aligned} (a, b) \in [\![P]\!](v) &\Leftrightarrow a \in v(A) \wedge a = b \\ &\Leftrightarrow (a, b) \lhd_{(A,B),v} Id_A \text{ by definition} \end{aligned} \tag{12}$$

Case $P = Chain(P', Q)$

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

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

Case $P = And(P', Q)$

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

Case $P = Or(P', Q)$

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

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

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

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

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

Case $P = Distinct(P')$

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

Case $P = Exactly(n, P')$

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

$$\text{let } f = (\lambda pairs.join([\![P']\!](v), pairs)) \tag{19}$$

then

$$(a, b) \in [\![P]\!](v) \Leftrightarrow (a, b) \in f^n[\![Id_A]\!](v) \tag{20}$$

hence it suffices to prove

$$(a, b) \in f^n[\![Id_A]\!](v) \Leftrightarrow (a, b) \lhd_{(A,A),v} Exactly(n, P') \tag{21}$$

Case $Exactly(0, P')$:

$$f^0[\![Id_A]\!](v) = [\![Id_A]\!](v)$$

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

$$\begin{aligned}(a, b) \in f^0[\![Id_A]\!](v) &\Leftrightarrow (a, b) \in [\![Id_A]\!](v)\\ &\Leftrightarrow (a, b) \lhd_{(A,A),v} Id_A\\ &\Leftrightarrow (a, b) \lhd_{(A,A),v} Exactly(0, P')\end{aligned} \tag{22}$$

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

$$f^{n+1}[\![Id_A]\!](v) = f(f^n([\![Id_A]\!](v)))$$

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

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

Case $P = Upto(n, P')$

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

$$\text{let } f = (\lambda pairs.join([\![P']\!](v), pairs) \cup pairs) \tag{24}$$

then

$$[\![P]\!](v) = f^n[\![Id_A]\!](v) \text{We now case split on n} \tag{25}$$

Case $Upto(0, P')$

$$\begin{aligned}(a, b) \in [\![Upto(0, P')]\!](v) &\Leftrightarrow (a, b) \in f^0[\![Id_A]\!](v)\\ &\Leftrightarrow (a, b) \in [\![Id_A]\!](v)\\ &\Leftrightarrow (a, b) \lhd_{(A,A),v} Id_A \text{ by } \Phi(\Sigma, Id_A, A, A)\\ &\Leftrightarrow (a, b) \lhd_{(A,A),v} Upto(0, P') \text{ by (Upto0)}\end{aligned} \tag{26}$$

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

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

case $P = FixedPoint(P')$

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

$$\begin{gathered}\text{let } f = (\lambda pairs.join([\![P']\!](v), pairs) \cup pairs) \\ \text{then} \\ [\![P]\!](v) = fix(f) \text{ In the domain } closure(A,v)\end{gathered} \tag{28}$$

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

Firstly, f is monotonous.

Let $x \subseteq y$

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

Secondly, f **preserves the** lub**s of chains.**

Consider a chain $x1 \subseteq x2 \subseteq ...$ in $closure(A,v)$ Since $closure(A,v)$ is a domain, the lub, $\bigcup_n x_n$ is also in $closure(A,v)$

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

To get the inverse relation,

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

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

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

So f is Scott-continuous.
Now, by Tarski's fixed point theorem

$$[\![FixedPoint(P')]\!](v) = fix(f) = \bigsqcup_n f^n(\bot)$$

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

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

Case $(a,b) \lhd_{A,A,v} Id_A$**:**
by $\Phi(\Sigma, Id_A, A, A), (a,b) \in [\![Id_A]\!](v) = \bot$
so $n = 0$

Case $(a,b) \lhd_{(A,A),v} P' \wedge (b,c) \lhd_{(A,A),v} FixedPoint(P')$
(hence $(a,c) \lhd_{(A,A),v} FixedPoint(P')$)
by $\Phi(\Sigma, P', A, A), and (b,c) \lhd_{(A,A),v} FixedPoint(P')$

$$(a,b) \in [\![P']\!](v) \wedge \exists n.(b,c) \in f^n(\bot)$$

Instantiating with $m = n$ gives

$$(a,b) \in [\![P']\!](v) \wedge (b,c) \in f^m(\bot)$$

so $(a,c) \in f(f^m(\bot)) = f^{m+1}(\bot)$
Hence $(a,b) \lhd_{(A,A),v} FixedPoint(P') \Rightarrow \exists n.(a,b) \in f^n(\bot)$

To go the other way, we need to prove $(a,b) \in \bigsqcup_n f^n(\bot) \Rightarrow (a,b) \lhd_{(A,A),v} FixedPoint(P')$
$(a,b) \in \bigsqcup_n f^n(\bot)$ Means that either:

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

$$\begin{aligned} &\therefore (a,b) \in [\![Id_A]\!](v) \\ &\therefore (a,b) \lhd_{(A,A),v} Id_A \text{ By } \Phi(\Sigma, Id_a, A, A) \\ &\therefore (a,b) \in [\![Id_A]\!](v) \text{ By (Fix1)} \end{aligned} \tag{33}$$

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

$$\begin{aligned}(\exists a'.(a,a') \in [\![P']\!](v) \wedge (a',b) \in f^n(\bot)) \vee (a,b) \in f^n(\bot) \wedge \neg((a,b) \in f^n(\bot)) \\ \therefore \exists a'.(a,a') \in [\![P']\!](v) \wedge (a',b) \in f^n(\bot) \\ \therefore (a,a') \lhd_{(A,A),v} P' \wedge (a',b) \lhd_{(A,A),v} FixedPoint(P') \\ \therefore (a,b) \lhd_{(A,A),v} FixedPoint(P')\end{aligned} \tag{34}$$

so we have $(a,b) \lhd_{(A,A),v} FixedPoint(P') \Leftrightarrow \exists n.(a,b) \in f^n(\bot) \Leftrightarrow (a,b) \in [\![FixedPoin(P')]\!](v)$

□

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.

$$write\colon View_\Sigma \to \wp(A \times R \times B) \rightharpoonup View_\Sigma \text{For } A, B \in \tau \tag{35}$$

We define *write* as so:
let $rs = \{(a_i, r_i, b_i) \in (A \times R \times B) \mid 0 < i \leqslant n\}$ for some n being the size of the set.

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