Translating Queries into Combinators

Mitch Cherniack
Brown University, Providence, RI 02912
Phone: (401) 863-7673 FAX: (401) 863-7657
email: mfc@cs.brown.edu

December 17, 1996

Abstract

In [CZ96], we argue that a combinator (i.e., variable-free) query algebra simplifies correctness proofs and implementations of query optimizers. But combinators make for poor query languages, and therefore translations from standard query languages (such as OQL [Cat93]) into some combinator form is required.

In this paper, we present our translator that maps queries written in a variant of OQL (OQL_{deB}) into KOLA [Che95a]; our combinator-based query algebra. We introduce a denotational semantics for OQL_{deB} and define KOLA using an operational semantics that rewrites KOLA queries into OQL_{deB}. We then proceed to show the correctness of our translator by showing that translation produces a KOLA query whose OQL_{deB} equivalent has the same semantics as the original query.

The significance of our work lies in its implications for the design of efficient and simply formalized query optimizers. With respect to formalization, we have used the Larch [GHG+92] specification tool LSL to formally specify KOLA and its associated rule language, and have used these specifications within the Larch theorem prover LP to verify the correctness of over 300 transformation rules. (For details of this work see [Che95a]). We were not the first to recognize the potential benefits of a combinator-based internal query algebra. But by introducing a translation technique to map OQL queries into their combinator forms, these benefits become realizable in practice.

1 Introduction

Query languages use variables to name and relate data objects held in collections. For example, the OQL [Cat93] query of Figure 1 pairs people in the set `P` with children who live with them. This query uses the variable x_0 to denote people in `P`, and x_1 to denote children of people in `P`. The predicate $(x_0 \to \texttt{addr}) == (x_1 \to \texttt{addr})$ expresses the relationship that holds between people and their children who live with them. OQL's `struct` construct permits the visualization of those `(Par, SAC)`[1] pairs satisfying the relationship.

Variables make queries easy to express but difficult to optimize. Because their meaning is tied to scope, variables obscure the recognition of optimizable classes of queries. Typically, an optimizer must perform

[1] (SAC is short for "Same Address Children")

$$\begin{array}{l}
\texttt{select distinct} \\
\quad \texttt{struct}\,(\texttt{Par}:\ x_0,\ \texttt{SAC}: \left(\begin{array}{l}\texttt{select distinct } x_1 \\ \texttt{from } x_1 \texttt{ in } (x_0 \to \texttt{child}) \\ \texttt{where } (x_0 \to \texttt{addr})\ ==\ (x_1 \to \texttt{addr})\end{array}\right)) \\
\texttt{from } x_0 \texttt{ in P}
\end{array}$$

Figure 1: An OQL Query With Variables x_0 and x_1

environment analysis to decide when optimizations can be applied [CZ96]. Environment analysis adds ineffiency to optimization and complicates formalization. Optimizers stand to benefit from having queries expressed in a variable-free *(combinator-based)* form. But combinators are not easy for query writers to use and so a translation into this form is required.

The problem addressed in this paper is to how to perform such translations. We introduce a technique that we have used to implement translators from both the AQUA [LMS+93] and OQL query languages into KOLA [CZ96]; our combinator-based algebra. We describe the technique by presenting the OQL $\rightarrow$ KOLA translator.

The significance of our work lies in its implications for query optimizer design. The existence of a combinator-based query algebra and a technique that translates queries into that form makes a combinator-based approach to optimization plausible. Thus, it should be possible for an optimizer to avoid the environment analysis that introduces inefficiencies and complicates formalization.

The paper proceeds as follows. In Section 2, we describe a variant of OQL (limiting it to queries that are over sets, tuples and objects), and give it a denotational semantics. In Section 3, we use the denotational semantics to prove identities of OQL expressions. In Section 4 we describe KOLA and give it an operational (reduction) semantics. We then describe our translation function **T**, that maps OQL expressions to their KOLA equivalents and show its correctness. In Section 5, we compare our work with related work in databases concerning querying with combinators, and related work in functional programming concerning translations into combinator forms.

2 A Denotational Semantics of OQL$_{deB}$

In this section, we describe our variant of the query language OQL [Cat93], which serves as the source language for translation. Hereafter we refer to our OQL variant as *OQL$_{deB}$* (or *deBruijn*-OQL).

We begin in Section 2.1 by describing how OQL$_{deB}$ differs from OQL. In short,

1. OQL$_{deB}$ includes constants $\diamond$ (the empty tuple) and $\{\}$ (the empty set),

2. OQL$_{deB}$ includes an `in _ then _ else _` expression,

3. OQL$_{deB}$ reserves "dot notation" ($Q \;.\; a_i$) for extracting *fields* of tuples, and "arrow notation" ($Q \rightarrow$ `att`) for invoking *methods* on objects,

4. OQL$_{deB}$ does not include support for:

   - floats, chars and strings (These are easy to add and will be added soon.)
   - bags, arrays and lists (These require extending KOLA. This is anticipated future work.) Note: Because sets are the only collection type supported in OQL$_{deB}$, all `select` queries will be written with `select distinct`, even though this may not be necessary in all cases.
   - queries that construct new *mutable* objects (The semantics of this is unclear, even in [Cat93]. This is anticipated future work.)
   - query naming, class indication, queries with the `having` operator, inclusion predicates and type casting (These should require straightforward extensions to the translator. This is anticipated future work.)

5. OQL$_{deB}$ assumes a normalizing scheme to simplify the translation of variables and tuples. This is described below.

Conversion of OQL queries into OQL_{deB} is straightforward and is described in Section 2.1. We provide a formal (denotational) semantics for OQL_{deB} in Section 2.2 and in the same section discuss the *well-formedness* property of meaningful OQL_{deB} queries.

2.1 OQL_{deB}: A Variant of OQL

2.1.1 Variable Normalization

OQL_{deB} deviates from the language definition of [Cat93] by assuming a deBruijn-like [DeB72] notation for all variables appearing within queries. Using this notation, variables are named uniformly (V_i for some $i \geq 0$), thus facilitating translation. As well, variable "declarations" in OQL `from` clauses (e.g., x_0 in "x_0 `in P`" and x_1 in "x_1 `in` ($x_0 \to$ `child`)" as in the query of Figure 1) are unnecessary in OQL_{deB} because variable declarations are implicit using deBruijn notation.

Intuitively, the uniformity of variable naming in a deBruijn scheme comes from the association of integer values with each variable reference. The value of i for a given variable reference, V_i is the "distance" from the reference to its declaration (measured by the number of variables that are in scope and that were declared after V_i was declared). V_0 therefore refers to the most "recently" declared variable (relative to the point in the parse where V_0 appears).

The translation of a query from OQL into OQL_{deB} is straightforward and resembles the deBruijn translation of the λ-calculus [DeB72]. The process is in two phases. Firstly, every subexpression is associated with a *scope list*; a list of all variables (from first to most recently declared) defined within the scope of the subexpression. Scope lists are propagated downward through the parse tree of the query. Secondly, translation proceeds from the leaves of the parse tree upwards. During this phase, each variable occurence is replaced by a deBruijn reference indicating the position of the variable within the scope list of the subexpression in which it appears.

To illustrate, we sketch the OQL $\to$ OQL_{deB} translation of OQL `select−distinct−from− where` set queries of the form shown in Figure 2. In this figure, Q_s, Q_w, and $Q_i, \ldots, Q_m$ $(0 \leq i \leq m)$ are OQL subexpressions, potentially including free occurences of variables. Suppose that this query appears as a subexpression of another query that has as its scope list,

$$(x_0, \ldots, x_{i-1}).$$

(In the case where the query is not a subexpression of another query, then this scope list would be empty.) Then,

1. for each `from` subexpression: Q_k $(i \leq k \leq m)$:
   - Q_k's scope list is $(x_0, \ldots, x_{k-1})$. (That is, the only variables that can appear *free* in Q_k are $x_0, \ldots,\ x_{k-1}$).
   - translation into OQL_{deB} replaces each x_j appearing in Q_k with V_{k-j-1}.
2. for `select distinct` subexpression, Q_s and `where` subexpression Q_w:
   - Q_s's and Q_w's scope lists are both $(x_0, \ldots, x_m)$.
   - translation into OQL_{deB} replaces each x_j in Q_s or Q_w with V_{m-j}.

Consider now, the OQL $\to$ OQL_{deB} translation of the OQL query of Figure 1. This query has no occurences of free variables and so is defined over an empty scope list. Therefore, translation proceeds as follows:

```
select distinct Q_s
from x_i in Q_i, ..., x_m in Q_m
where Q_w
```

Figure 2: A Prototypical OQL Set Query

1. *Outer* `from` *Clause:* "x_0 `in P`"

   This expression contains no free variable occurences (`P` is a global name and is therefore a constant rather than a variable). Translation therefore returns the expression `P`.

2. *Outer* `select distinct` *Clause:*

$$\text{“}\texttt{struct (Par}: x_0,\ \texttt{SAC}: \left(\begin{array}{l} \texttt{select distinct } x_1 \\ \texttt{from } x_1 \texttt{ in } (x_0 \rightarrow \texttt{child}) \\ \texttt{where } (x_0 \rightarrow \texttt{addr}) \ == \ (x_1 \rightarrow \texttt{addr}) \end{array} \right))\text{”}$$

   Translation of this expression is done within the context of the scope list, (x_0), created by the outer `from` clause. It proceeds as follows:

   - *Inner* `from` *Clause:* "x_1 `in` $(x_0 \rightarrow$ `child`$)$"

     This expression includes the reference to the free variable x_0. Using the translation scheme described, $k = 1$, $j = 0$ and translation returns the expression, $(V_0 \rightarrow \texttt{child})$.

   - *Inner* `select distinct` *Clause:* "x_1"

     Translation of this expression is done within the context of the scope list, (x_0, x_1). Using the translation scheme, $m = 1$, $j = 1$ and translation returns the expression, V_0.

   - *Inner* `where` *Clause:* "$(x_0 \rightarrow \texttt{addr}) \ == \ (x_1 \rightarrow \texttt{addr})$"

     Translation of this expression is done within the context of the scope list, (x_0, x_1). Using the translation scheme, $m = 1$, $j = 0$ (in "$x_0 \rightarrow \texttt{addr}$") and $j = 1$ (in "$x_1 \rightarrow \texttt{addr}$") and translation returns the expression, $(V_1 \rightarrow \texttt{addr}) \ == \ (V_0 \rightarrow \texttt{addr})$.

Thus, translation returns:

$$\begin{array}{l} \texttt{select distinct} \\ \quad \texttt{struct (Par}: V_0,\ \texttt{SAC}: \left(\begin{array}{l} \texttt{select distinct } V_0 \\ \texttt{from } (V_0 \rightarrow \texttt{child}) \\ \texttt{where } (V_1 \rightarrow \texttt{addr}) \ == \ (V_0 \rightarrow \texttt{addr}) \end{array} \right)) \\ \texttt{from P} \end{array}$$

Note that the relative referencing of deBruijn notation means that the same variable can have multiple meanings within a single query. In the above query, the variable V_0 refers to both an arbitrary person in `P` (everywhere but in the inner `select distinct` and `where` clauses) and also to an arbitrary child of some person in `P` (in the inner `select distinct` and `where` clauses).

2.1.2 Tuple Normalization

The other deviation from OQL assumed by OQL_{deB} is that tuples (`struct`s) are normalized. The goal of tuple normalization is to standardize tuples making them amenable to translation. Normalization restructures tuples so that they are always nested $\ll\gg$-delimited pairs having the form,

$$\ll a_0 : \ldots \ \ll a_0 : \ \diamond, \ a_1 : Q_0 \gg, \ldots, \ a_1 : Q_m \gg$$

($\diamond$ is the "empty" tuple). This standardizes the length of a tuple (always two[2]), the tuple's fieldnames (always a_0 and a_1), and the fieldnames that refer to values held in the tuple (always a_1). This standardization simplifies both the language's semantics and its translation into KOLA.

The normalization of any tuple expression,

$$\texttt{struct}\ (<label_0>: Q_0, \ldots,\ <label_m>: Q_m)$$

proceeds in two steps:

1. fields are reordered according to some predefined ordering (e.g., a lexicographical ordering on fieldnames). The result is an equivalent struct expression with the fields permuted:

$$\texttt{struct}\ (<label_{p(0)}>: Q_{p(0)}, \ldots,\ <label_{p(m)}>: Q_{p(m)})$$

2. the permuted tuple is then restructured into the nested pair:

$$\ll a_0 : \ldots \ll a_0 :\ \diamond,\ a_1 : Q_{p(0)} \gg, \ldots,\ a_1 : Q_{p(m)} \gg.$$

   From here on in, we write:

$$\ll Q_0, \ldots, Q_n \gg$$

   as shorthand for the nested pair:

$$\ll a_0 : \ldots \ll a_0 :\ Q_0,\ a_1 : Q_1 \gg, \ldots,\ Q_n \gg$$

   for $n \geq 0$. ($\ll Q_0, \ldots, Q_0 \gg$ is equivalent to Q_0.) Thus, restructuring of the permuted tuple produces:

$$\ll \diamond,\ Q_{p(0)}, \ldots,\ Q_{p(m)} \gg.$$

To illustrate normalization, consider again the query of Figure 1. Earlier, we showed the conversion of this query into deBruijn notation. Tuple normalization transforms this form into OQL$_{deB}$:

$$\begin{array}{l} \texttt{select distinct} \\ \quad \ll \diamond,\ V_0, \left(\begin{array}{l} \texttt{select distinct}\ V_0 \\ \texttt{from}\ (V_0 \rightarrow \texttt{child}) \\ \texttt{where}\ (V_1 \rightarrow \texttt{addr})\ ==\ (V_0 \rightarrow \texttt{addr}) \end{array} \right) \gg. \\ \texttt{from P} \end{array}$$

Queries that *extract* the values of fields from tuples also have to be altered in accordance with the restructuring of tuples into nested pairs. Every "field extraction" expression will be of the form described by the expression below:

$$Q\ .\ a_0^i\ .\ a_1$$

such that Q is an expression that evaluates to a tuple, and

$$\begin{array}{lcl} Q\ .\ a_0^0\ .\ a_1 & = & Q\ .\ a_1 \\ Q\ .\ a_0^1\ .\ a_1 & = & Q\ .\ a_0\ .\ a_1 \\ & \cdots & \\ Q\ .\ a_0^i\ .\ a_1 & = & Q\ .\ \underbrace{a_0\ .\ \ldots\ .\ a_0}_{i}\ .\ a_1. \end{array}$$

[2] Even an OQL tuple with only one field will be normalized into a pair, with the additional field a_0 given the value, $\diamond$.

Thus, $Q\ .\ a_1$ extracts the last value of the tuple, $Q\ .\ a_0\ .\ a_1$, the second-to-last and so on. To illustrate, the OQL query:

$$
\begin{array}{l}
\texttt{select distinct } x_2\ .\ \texttt{SAC} \\
\texttt{from } x_2 \texttt{ in } \left(\begin{array}{l}
\texttt{select distinct} \\
\quad \texttt{struct (Par : } x_0,\ \texttt{SAC : } \left(\begin{array}{l}
\texttt{select distinct } x_1 \\
\texttt{from } x_1 \texttt{ in } (x_0 \rightarrow \texttt{child}) \\
\texttt{where } (x_0 \rightarrow \texttt{addr}) \ == \ (x_1 \rightarrow \texttt{addr})
\end{array} \right) \texttt{)} \\
\texttt{from } x_0 \texttt{ in P}
\end{array} \right) \\
\texttt{where } x_2\ .\ \texttt{Par}\ .\ \texttt{age} \ > 25
\end{array}
$$

which extracts from fields `Par` and `SAC` of tuples named by x_2 would be translated into the OQL$_{deB}$ query:

$$
\begin{array}{l}
\texttt{select distinct } V_0\ .\ a_1 \\
\texttt{from } \left(\begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0,\ \left(\begin{array}{l}
\texttt{select distinct } V_0 \\
\texttt{from } (V_0 \rightarrow \texttt{child}) \\
\texttt{where } (V_1 \rightarrow \texttt{addr}) \ == \ (V_0 \rightarrow \texttt{addr})
\end{array} \right) \gg \\
\texttt{from P}
\end{array} \right) \\
\texttt{where } ((V_0\ .\ a_0\ .\ a_1) \rightarrow \texttt{age}) \ > 25.
\end{array}
$$

It is important to remember that the tuple expression that describes the (parent, set of children) pair is shorthand for a nested pair expression, and therefore field extraction subexpressions $V_0\ .\ a_1$ (in the `select` clause) and $V_0\ .\ a_0\ .\ a_1$ (in the `where` clause) are meaningful.

Our proposed tuple normalization is "lossy" in one respect: it is possible to construct OQL queries that are not type-correct but that can *become* type-correct once translated into OQL$_{deB}$. A simple example is,

```
set (struct (a: 3, b: 5)) union set (struct (c: 4, d: 6))
```

which is not type correct because of the incompatible fieldnames of tuples contained in the sets being unioned. However, the OQL$_{deB}$ translation of this query:

$$\texttt{set } (\ll \diamond,\ \texttt{3, 5} \gg) \texttt{ union set } (\ll \diamond,\ \texttt{4, 6} \gg)$$

is type-correct. We avoid the difficulty that this introduces by assuming that all OQL queries subject to translation are type-correct. In practice, the user would pose a query in OQL, have it examined by a type-checker, and (if type-checking succeeds) have it translated into OQL$_{deB}$ and then KOLA.[3]

This solution does not solve the type-checking problem if it is possible to name the sets produced by querying. That is, suppose we name the sets,

```
A  =  set (struct (a: 3, b: 5)), and
B  =  set (struct (c: 4, d: 6))
```

and then proceed to union `A` and `B`. Unless the fieldnames of the original queries are preserved, the union of `A` and `B` will mistakenly be considered type-safe because the types of `A` and `B` will be OQL$_{deB}$ types and hence lacking tuple fieldnames. We propose a simple hack to solve this problem; our implemented translator generates an auxiliary table used to map any tuples produced by the query back into tuples with fieldnames. Thus, tuples are left without fieldnames only for the duration of a query's evaluation.

2.2 Semantics of OQL$_{deB}$

In this section, we present a formal (denotational) semantics for OQL$_{deB}$. This semantics provide not only a precise definition of the language, but also allow us to prove that certain OQL$_{deB}$ expressions are equal, and also to prove that our translation of OQL$_{deB}$ into KOLA is correct.

[3] In fact our translator [Che96] skips the translation into OQL$_{deB}$ to make translation into KOLA more efficient.

2.2.1 $\mathcal{D}$: A Semantic Domain

To give a semantics to OQL$_{deB}$ expressions we assume a semantic domain, $\mathcal{D}$ that includes base values (*integers* and *booleans*), objects (whose interface is determined by their *class*), pairs and sets. The operators that can be applied to objects depend on their type interfaces. The operators over all other values are standard ones and are listed below. Typewriter style distinguishes OQL$_{deB}$ constants (such as `3`) from values in $\mathcal{D}$ (e.g., 3).

- *Booleans*: have constructors *true* and *false* and the operators $=$, $\neg$, $\wedge$, and $\vee$ which have their usual meanings.
- *Integers*: have constructors $0,\ 1,\ -1,\ \ldots$ and operators $+$, $\times$, $-$ (subtraction), $\div$, $-$ (inverse), $|\ |$ (absolute value), $\%$ (modulo), $<$, $>$, $\leq$, $\geq$ and $=$ which have their usual meanings.
- *Objects*: are assumed to have an interface provided to a schema in advance (via a class definition). For example, the class of `Person` objects might include attributes *addr* which returns a person's address, *age* which returns a person's age, and *child* which returns the set of `Person`s who are children of a given person. For simplicity, we assume here that all methods defined over a particular object type are unary.
- *Pair*: has constructors ϵ (the empty pair) and $(\_,\ \_)$, and operator $=$ which has its usual meanings. We use the notation,

  $$(d_0,\ d_1, \ldots,\ d_n)$$

  for $d_i \in \mathcal{D}$, as shorthand for the nested pair expression,

  $$(\ldots(d_0,\ d_1), \ldots,\ d_n).$$

  Any such expression with $d_0 = \epsilon$ is an *environment* (see Section 2.2.2).
- *Set*: has constructors $\oslash$ and $\{\ldots\}$ and operators $\in$, $\cup$, $\cap$, $-$, $|\ |$ (cardinality) and $=$. We restrict $\mathcal{D}$ to include only homogeneous sets.

2.2.2 A Denotational Semantics of OQL$_{deB}$

The "meaning" of an OQL$_{deB}$ expression is dependent on an *environment*; a nested pair value in $\mathcal{D}$ of the form,

$$\rho = (\epsilon,\ d_0, \ldots,\ d_n).$$

An environment associates values in $\mathcal{D}$ with all variable references appearing in the expression. We adopt the traditional approach to denotational semantics and associate every OQL$_{deB}$ expression Q with a function (**Eval** $[\![Q]\!]$) that maps environments to values in $\mathcal{D}$.

The semantics of OQL$_{deB}$ expressions is shown in Figure 4. Most of the equations are straightforward, but two deserve special mention. The semantics of `select−distinct−from−where` and `select−distinct−from−where − groupby` queries are such that the environments accessed by subexpressions of these queries are not uniform. To understand the intuition behind the semantics of these queries, it is helpful to consider the semantics of the queries' subexpressions. Given a query,

$$\begin{array}{l}\texttt{select distinct } Q_s\\ \texttt{from } Q_0, \ldots,\ Q_m\\ \texttt{where } Q_w\end{array}$$

its semantics wrt some environment ρ is determined as follows:

1. first, the semantics of subexpression Q_0 is determined wrt ρ (**Eval** $[\![Q_0]\!]\ \rho$). This should result in a set.

2. next, any element $x_0 \in$ **Eval** $[\![Q_0]\!]\ \rho$ is selected and the semantics of subexpression Q_1 is evaluated wrt $(\rho,\ x_0)$. Then, some x_1 is selected from this set and Q_2 is evaluated wrt $(\rho,\ x_0,\ x_1)$ and so on. This captures the dependencies between `from` sets that are permitted in OQL.

3. after processing each of $Q_0, \ldots,\ Q_m$ in this fashion, a set of environments, $(\rho,\ x_0, \ldots,\ x_m)$ is constructed where each x_i is an element of the set **Eval** $[\![Q_i]\!]\ (\rho,\ x_0, \ldots,\ x_{i-1})$. For any such environment that makes Q_w evaluate to *true*, the result of evaluating Q_s with that environment is added to the query's set result.

"Group-by" queries such as

select distinct Q_s
from $Q_0, \ldots,\ Q_m$
where Q_w
groupby $Q_{m+1}, \ldots,\ Q_p$,

have much the same flavor. Each of the subexpressions $Q_{m+1}, \ldots,\ Q_p$ is expected to be a *predicate* (i.e., evaluate to either *true* or *false*). Thus, evaluation of this query proceeds as follows:

1. first, the semantics of

   select distinct Q_s
   from $Q_0, \ldots,\ Q_m$,
   where Q_w

   is determined wrt ρ in the manner described previously. This results in a set, R.

2. next, the elements of R are grouped according to the evaluation of each predicate $Q_{m+1}, \ldots,\ Q_p$. That is, if $x \in R$, then the partition of R containing x will consist of all elements of R, y such that Q_{m+1} evaluates to the same result wrt $(\rho,\ x)$ and $(\rho,\ y)$, Q_{m+2} evaluates to the same result wrt $(\rho,\ x)$ and $(\rho,\ y)$, and so on up to and including the results of evaluating Q_p wrt $(\rho,\ x)$ and $(\rho,\ y)$.

The evaluation of this query results in a set of environment values,

$$(\epsilon,\ r_{m+1}, \ldots, r_p,\ f\ (x))$$

for each $x \in R$ where $f(x)$ is the partition of x just described, and each value r_i is either 1 or 0, depending on the result (*true* or *false*) of evaluating Q_i wrt $(\rho,\ x)$.

2.2.3 Well-Formedness of OQL$_{deB}$ Queries

Not every OQL$_{deB}$ expression built according to the grammar of Figure 3 is well-formed. For example, the expression "`3 + T`" is not well-formed because both arguments to + must be integers. The expression "V_0" is not well-formed because variable references can only appear within `select−distinct` queries.

Definition 2.1 (Well-formedness wrt environments) *An OQL$_{deB}$ expression Q is well-formed wrt environment ρ if the semantics of Q wrt ρ (**Eval** $[\![Q]\!]\ \rho$) is defined according to the rules of Figure 4.*

Intuitively, the well-formedness of Q wrt ρ is established by type-checks and variable checks. Type checks ensure that OQL$_{deB}$ operations are only invoked on their appropriate operands. For example, both arguments to + must be expressions that evaluate to integers while the first argument to `if _ then _ else _` must be an expression that evaluates to a boolean. Similarly, all expressions of the form $Q\ \rightarrow$ `att` must be

$$
\begin{array}{rcl}
Q & :: & c \quad (\text{where } c \text{ is any of } \texttt{()},\ \{\},\ \texttt{0},\ \texttt{1},\ \texttt{-1},\ \ldots,\ \texttt{T},\ \texttt{F},\ id\ (id \text{ a global name}),\ \ldots) \\
& | & V_i \quad (\text{where } i \geq 0) \\
& | & \ll Q, \ldots, Q \gg \\
& | & Q\ \texttt{.}\ a_0 \\
& | & Q\ \texttt{.}\ a_1 \\
& | & Q \rightarrow \texttt{att} \quad (att \text{ an attribute on the type of } Q) \\
& | & -\, Q \\
& | & \texttt{abs}\ (Q) \\
& | & Q\ \ ArithOp\ \ Q \\
& | & Q\ \ SetOp\ \ Q \\
& | & \texttt{set}\ (Q, \ldots, Q) \\
& | & \texttt{element}\ (Q) \\
& | & \texttt{flatten}\ (Q) \\
& | & Agg\ (Q) \\
& | & \texttt{select distinct}\ Q\ \texttt{from}\ Q, \ldots, Q\ \texttt{where}\ Q \\
& | & \texttt{select distinct}\ Q\ \texttt{from}\ Q, \ldots, Q\ \texttt{where}\ Q\ \texttt{groupby}\ Q, \ \ldots\ Q \\
& | & \texttt{if}\ Q\ \texttt{then}\ Q\ \texttt{else}\ Q \\
& | & (Q) \\
& | & Q\ \ CompOp\ \ Q \\
& | & Q\ \texttt{in}\ Q \\
& | & \texttt{not}\ (Q) \\
& | & Q\ \ BoolOp\ \ Q \\
& | & \texttt{for\_all}\ Q\ :\ Q \\
& | & \texttt{exists}\ Q\ :\ Q \\
& | & \texttt{exists}\ Q \\
& | & \texttt{unique}\ Q \\
& | & Q\ \ CompOp\ \ \texttt{some}\ (Q) \\
& | & Q\ \ CompOp\ \ \texttt{all}\ (Q) \\
ArithOp & :: & + \mid - \mid * \mid / \mid \texttt{mod} \mid \\
CompOp & :: & == \mid != \mid > \mid < \mid \geq \mid \leq \\
SetOp & :: & \texttt{intersect} \mid \texttt{union} \mid \texttt{except} \\
BoolOp & :: & \texttt{and} \mid \texttt{or} \\
Agg & :: & \texttt{count} \mid \texttt{sum} \mid \texttt{avg} \mid \texttt{min} \mid \texttt{max}
\end{array}
$$

Figure 3: The Syntax of an OQL_{deB} Expression, Q

justified by the underlying schema. (The type of Q must include an attribute named *att*.) Variable checks ensure that ρ is a large enough environment to ensure that every variable reference V_i that appears as a subexpression of Q can be evaluated via **Eval**. By our assumptions stated in Section 2.1.2, type-checking and hence variable-checking is done of OQL queries before their translation into OQL_{deB}.

The well-formedness of a query is a special case of well-formedness wrt environments.

Definition 2.2 (Well-formed queries (*wfq*'s)) *OQL_{deB} expression Q is a well-formed query (wfq) if it is well-formed wrt ϵ.*

The following lemma shows that the semantics of a *wfq* is independent of environments.

Lemma 2.1 (Well-formed query semantics) *Given wfq Q, and environment ρ*

$$\textbf{Eval}\ \ [\![Q]\!]\ \rho \quad = \quad \textbf{Eval}\ \ [\![Q]\!]\ \epsilon.$$

Proof: The proof follows from the observation that a *wfq*, because it is well-formed wrt ϵ, cannot reference any members of ρ and therefore has meaning independent of ρ. □

Given that the semantics of a *wfq* Q is a constant function, we can express the semantics of *wfq*'s as values in $\mathcal{D}$ rather than functions over environments. Therefore, throughout the rest of the paper we refer to the value of **Eval** $[\![Q]\!]\ \epsilon$ for the *wfq*, Q as "the semantics of Q".

3 Identities of $\textbf{OQL}_{deB}$ Expressions

In this section, we present identities expressed over OQL_{deB} and proven in terms of its denotational semantics. Many of these identities require altering an OQL_{deB} expression to change how the expression accesses its environment. We therefore begin by introducing notation that uniformly changes deBruijn indices of variables appearing in OQL_{deB} expressions.

3.1 Environment Reference Manipulation

Consider the example OQL_{deB} query below. In this query, `A` and `B` are sets. Objects in `A` have attributes g, h and k while objects in `B` have attributes f and j.

$$\begin{array}{l}\texttt{select distinct } (V_0 \to g) \\ \texttt{from } A \\ \texttt{where } (V_0 \to h) \texttt{ in } \left(\begin{array}{l}\texttt{select distinct } (V_0 \to f) \\ \texttt{from } B \\ \texttt{where } (V_1 \to k) == (V_0 \to j)\end{array}\right).\end{array}$$

Kim [Kim82] identified this query as a *Type J* nested query and illustrated that it was subject to a transformation into the form shown below:

$$\begin{array}{l}\texttt{select distinct } (V_1 \to g) \\ \texttt{from } A,\ B \\ \texttt{where } ((V_1 \to h) \ == \ (V_0 \to f)) \texttt{ and } ((V_1 \to k) \ == \ (V_0 \to j))\end{array}$$

This transformation is useful because it rewrites a query that demands evaluation via a nested loop strategy into a join query that can be evaluated using various join techniques.

The repositioning of subexpressions in the transformed query above requires an adjustment of deBruijn indices. For example, in the original query the extraction of attributes g and h from elements of set `A` is accomplished via subexpressions "$(V_0 \to g)$" and "$(V_0 \to h)$" respectively. The corresponding subexpressions in the transformed query are "$(V_1 \rightarrow g)$" and "$(V_1 \rightarrow h)$". In other words, the deBruijn indices of some variable references in the initial query are incremented by one as a result of transformation. This is because the environments associated with the initial query's `select` and `where` clause subexpressions differ from the corresponding subexpressions in the transformed query. In the initial query, only members of `A` appear in these environments and therefore, V_0 references members of this set. In the transformed query, members of both `A` and `B` appear in the corresponding subexpression environments and therefore, references to elements of `A` require the deBruijn variable, V_1.

To express adjustments to deBruijn indices in arbitrary expressions, we introduce the expression qualifiers, $Q \uparrow \binom{k}{j}$ (the *abstraction* of OQL_{deB} expression Q) and $Q \downarrow \binom{Q'}{j}$ (the *application* of OQL_{deB} expression Q' to OQL_{deB} expression Q). Roughly, $Q \uparrow \binom{k}{j}$ increments the deBruijn indices of variables $V_j,\ V_{j+1},\ V_{j+2}, \ldots$ appearing in Q, and $Q \downarrow \binom{Q'}{j}$ decrements the deBruijn indices of variables $V_{j+1},\ V_{j+2}, \ldots$ while replacing occurences of V_j with Q. In fact this is an oversimplification and not entirely accurate – formal definitions of these qualifiers appear in the following subsections.

3.1.1 Abstraction:

The OQL_{deB} expression $Q \uparrow \binom{k}{j}$ denotes an OQL_{deB} expression that resembles Q but which is well-formed for an environment that has k more members than any for which Q is well-formed, with all k members inserted beginning in the j^{th} position. Therefore, this expression denotes the *abstraction* of Q from an expression requiring instantiation in n arguments to one requiring instantiation in $n+k$ arguments. In particular, if Q is well-formed wrt $\rho = (\epsilon,\ x_n, \ldots,\ x_j, \ldots,\ x_0)$, then each variable reference in Q, V_i that evaluates to any of $x_n, \ldots,\ x_j$ is replaced by V_{i+k} in $Q \uparrow \binom{k}{j}$. This leads to the following identity proven in Lemma 3.1: Given OQL_{deB} expression Q that is well-formed wrt

$$(\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)$$

and values $d_1, \ldots,\ d_k \in \mathcal{D}$,

$$\textbf{Eval}\ [\![Q \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \quad = \quad \textbf{Eval}\ [\![Q]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0).$$

For simple OQL_{deB} expressions, all references to environment values $x_j, \ldots,\ x_n$ are by corresponding variable references $V_j, \ldots,\ V_n$. Therefore, we have such trivial identities as

$$\begin{array}{l} V_3 \uparrow \binom{2}{3} = V_5, \\ V_4 \uparrow \binom{2}{3} = V_6 \text{ and} \\ V_2 \uparrow \binom{2}{3} = V_2. \end{array}$$

But because subexpression environments are not uniform in OQL_{deB} set query expressions, abstraction of set query expressions is more complicated. In general,

$$Q \quad = \quad \left(\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0, \ldots,\ Q_m \\ \texttt{where } Q_w \end{array} \right) \uparrow \tbinom{k}{j} \quad = \quad \begin{array}{l} \texttt{select distinct } Q_s \uparrow \binom{k}{j+m+1} \\ \texttt{from } Q_0 \uparrow \binom{k}{j}, \ldots,\ Q_m \uparrow \binom{k}{j+m} \\ \texttt{where } Q_w \uparrow \binom{k}{j+m+1}. \end{array}$$

The varying values for v in $Q \uparrow \binom{k}{j+v}$ for each subexpression of Q, reflects the differing environments associated with subexpressions of Q. Namely, given that Q is well-formed wrt $(\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)$:

- each `from` subexpression Q_i $(0 \leq i \leq m)$ is well-formed wrt $(\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0,\ y_0, \ldots,\ y_{i-1})$, and
- `select` subexpression Q_s, and `where` subexpression Q_w are well-formed wrt

  $$(\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0,\ y_0, \ldots,\ y_m)$$

  such that each y_i is an element of the set denoted by Q_i.

Environment manipulation must therefore vary from subexpression to subexpression. For subexpressions Q_s and Q_w, variable V_{j+m+1} references environment value x_j. Thus, Q_s and Q_w should be replaced with $Q_s \uparrow \binom{k}{j+m+1}$ and $Q_w \uparrow \binom{k}{j+m+1}$ respectively. For each subexpression Q_i $(0 \leq i \leq m)$, variable V_{j+i} references environment value x_j. Thus, each Q_i should be replaced with $Q_i \uparrow \binom{k}{j+i}$.

3.1.2 Application:

The OQL_{deB} expression $Q \downarrow \binom{Q'}{j}$ denotes an OQL_{deB} expression that resembles Q but which fixes one of its variable references to be the OQL_{deB} expression Q'. Therefore, this expression denotes the *curried application* of Q' to one of the variables of Q. In particular, if Q is well-formed wrt $\rho = (\epsilon,\ x_n, \ldots,\ x_j, \ldots,\ x_0)$, then $Q \downarrow \binom{Q'}{j}$ is constructed as follows:

- each variable reference in Q that evaluates to x_j is replaced by Q', and
- each variable reference in Q, V_k that evaluates to any of $x_n, \ldots,\ x_{j+1}$ is replaced by V_{k-1}.

Note that the latter point means that application is not simply a matter of *substitution.* Application also requires accounting for shrunken environments by decrementing deBruijn indices. This leads to the following identity proven in Lemma 3.2: Given environment $\rho = (\epsilon,\ x_n, \ldots,\ x_{j+1},\ x_{j-1}, \ldots,\ x_0)$ (each $x_i \in \mathcal{D}$), OQL_{deB} expression Q' that is well-formed wrt ρ and Q that is well-formed wrt

$$(\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0),$$

then

$$\textbf{Eval}\ [\![Q \downarrow \tbinom{Q'}{j}]\!]\ \rho\ =\ \textbf{Eval}\ [\![Q]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0).$$

For simple OQL_{deB} expressions, all references to environment value x_j are by variable reference V_j. For example,

$$\begin{array}{l} V_3 \downarrow \binom{Q'}{3} = Q', \\ V_4 \downarrow \binom{Q'}{3} = V_3 \text{ and} \\ V_2 \downarrow \binom{Q'}{3} = V_2. \end{array}$$

But as with abstraction, application of set query expressions is more complicated. In general,

$$Q \ = \ \left(\begin{array}{l} \texttt{select distinct}\ Q_s \\ \texttt{from}\ Q_0, \ldots,\ Q_m \\ \texttt{where}\ Q_w \end{array} \right) \downarrow \binom{Q'}{j} \ = \ \begin{array}{l} \texttt{select distinct}\ Q_s \downarrow \binom{(Q' \uparrow \binom{m+1}{0})}{j+m+1} \\ \texttt{from}\ Q_0 \downarrow \binom{Q'}{j}, \ldots,\ Q_m \downarrow \binom{(Q' \uparrow \binom{m}{0})}{j+m} \\ \texttt{where}\ Q_w \downarrow \binom{(Q' \uparrow \binom{m+1}{0})}{j+m+1}. \end{array}$$

The varying values for v in $Q \downarrow \binom{(Q' \uparrow \binom{v}{0})}{j+v}$ for each subexpression of Q, reflects the differing environments associated with subexpressions of Q. Namely, given that Q is well-formed wrt $(\epsilon,\ x_n, \ldots,\ x_j, \ldots,\ x_0)$:

- each `from` subexpression Q_i $(0 \leq i \leq m)$ is well-formed wrt $(\epsilon,\ x_n, \ldots,\ x_j, \ldots,\ x_0,\ y_0, \ldots,\ y_{i-1})$, and
- `select` subexpression Q_s, and `where` subexpression Q_w are well-formed wrt $(\epsilon,\ x_n, \ldots,\ x_j, \ldots,\ x_0,\ y_0, \ldots,\ y_m)$ such that each y_i is an element of the set denoted by Q_i.

Environment manipulation must therefore vary from subexpression to subexpression. For subexpressions Q_s and Q_w:

- Expression Q' must ignore environment values $y_0, \ldots,\ y_m$. Therefore, references that evaluate to x_j must be replaced with $Q' \uparrow \binom{m+1}{0}$.
- Variable V_{j+m+1} references environment value x_j. Thus, Q_s and Q_w should be replaced with $Q_s \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1}$ and $Q_w \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1}$ respectively.

For each subexpression Q_i $(0 \leq i \leq m)$:

- Expression Q' must ignore environment values $y_0, \ldots, y_{i-1}$. Therefore, references that evaluate to x_j must be replaced with $Q' \uparrow \binom{i}{0}$.
- Variable V_{j+i} references environment value x_j. Thus, Q_i should be replaced with $Q_i \downarrow \binom{Q' \uparrow \binom{i}{0}}{j+i}$.

Formal definitions of $Q \uparrow \binom{n}{j}$ and $Q \downarrow \binom{Q'}{j}$ for all OQL expressions Q are defined in terms of the OQL grammar in Figure 5.

3.2 Identities

3.2.1 Environment Identities

Below are some identities of expressions that are environment-dependent.

Lemma 3.1 *For any OQL expression, Q, that is well-formed wrt environment $\rho = (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)$ and any values $d_1, \ldots,\ d_k \in \mathcal{D}$:*

$$\textbf{Eval}\ \ [\![Q \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \ = \ \textbf{Eval}\ \ [\![Q]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)$$

Proof: The proof is by structural induction on Q.

Base Case: For the base cases we have

Case: $Q \ \equiv\ c$ (for constant value, c).

$$\begin{aligned}
\textbf{Eval}\ \ [\![c \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \ &= \ \textbf{Eval}\ \ [\![c]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&= \ \textbf{Val}\ [\![c]\!] \\
&= \ \textbf{Eval}\ \ [\![c]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}$$

Case: $Q \ \equiv\ V_i$ (for $0 \leq i \leq n$)

$$\begin{aligned}
&\textbf{Eval}\ \ [\![V_i \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&= \begin{cases} \textbf{Eval}\ \ [\![V_{i+k}]\!]\ (\epsilon,\ x_n, \ldots,\ x_i, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0), & \text{if } i \geq j, \\ \textbf{Eval}\ \ [\![V_i]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_i, \ldots,\ x_0), & \text{if } i < j \end{cases} \\
&= \ x_i \\
&= \begin{cases} \textbf{Eval}\ \ [\![V_i]\!]\ (\epsilon,\ x_n, \ldots,\ x_i, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0), & \text{if } i \geq j, \\ \textbf{Eval}\ \ [\![V_i]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_i, \ldots,\ x_0), & \text{if } i < j \end{cases} \\
&= \ \textbf{Eval}\ \ [\![V_i]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}$$

Inductive Case: The inductive case must be proved by cases that vary according to the structure of Q.

Case: $Q \ \equiv\ \ll Q_0, \ldots,\ Q_m \gg$

$$\begin{aligned}
&\textbf{Eval}\ \ [\![\ll Q_0, \ldots,\ Q_m \gg \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&= \ \textbf{Eval}\ \ [\![\ll Q_0 \uparrow \tbinom{k}{j}, \ldots,\ Q_m \uparrow \tbinom{k}{j} \gg]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&= \ \begin{array}{l} (\textbf{Eval}\ \ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0), \ldots, \\ \ \textbf{Eval}\ \ [\![Q_m \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)) \end{array} \\
&= \ (\textbf{Eval}\ \ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0), \ldots,\ \textbf{Eval}\ \ [\![Q_m]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)) \ \ \text{(by induction)} \\
&= \ \textbf{Eval}\ \ [\![\ll Q_0, \ldots,\ Q_m \gg]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}$$

Case: $Q \equiv Q_0 \,.\, a_0$ (if **Eval** $[\![Q_0]\!]\,\rho = (\delta,\ v)$)

$$\begin{aligned}
&\textbf{Eval}\ [\![(Q_0\ .\ a_0)\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ \textbf{Eval}\ [\![(Q_0\uparrow\tbinom{k}{j})\ .\ a_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ \delta,\quad (\text{if } \textbf{Eval}\ [\![Q_0\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0) = (\delta,\ v))\\
&\quad=\ \delta,\quad (\text{if } \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0) = (\delta,\ v))\ \ (\text{by induction})\\
&\quad=\ \textbf{Eval}\ [\![Q_0\ .\ a_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q \equiv Q_0 \,.\, a_1$ (if **Eval** $[\![Q_0]\!]\,\rho = (\delta,\ v)$)

$$\begin{aligned}
&\textbf{Eval}\ [\![(Q_0\ .\ a_1)\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ \textbf{Eval}\ [\![(Q_0\uparrow\tbinom{k}{j})\ .\ a_1]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ v,\quad (\text{if } \textbf{Eval}\ [\![Q_0\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0) = (\delta,\ v))\\
&\quad=\ v,\quad (\text{if } \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0) = (\delta,\ v))\ \ (\text{by induction})\\
&\quad=\ \textbf{Eval}\ [\![Q_0\ .\ a_1]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q \equiv Q_0 \rightarrow \texttt{att}$ (if *att* is defined for the type of Q_0)

$$\begin{aligned}
&\textbf{Eval}\ [\![(Q_0\rightarrow\ \texttt{att})\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ \textbf{Eval}\ [\![(Q_0\uparrow\tbinom{k}{j})\rightarrow\ \texttt{att}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ \mathit{att}\ (\textbf{Eval}\ [\![Q_0\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0))\\
&\quad=\ \mathit{att}\ (\textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0))\ \ (\text{by induction})\\
&\quad=\ \textbf{Eval}\ [\![Q_0\rightarrow\ \texttt{att}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q \equiv -\ Q_0$

$$\begin{aligned}
&\textbf{Eval}\ [\![(-\ Q_0)\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ \textbf{Eval}\ [\![-(Q_0\uparrow\tbinom{k}{j})]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ -\ (\textbf{Eval}\ [\![Q_0\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0))\\
&\quad=\ -\ (\textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0))\ \ (\text{by induction})\\
&\quad=\ \textbf{Eval}\ [\![-\ Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q \equiv \texttt{abs}\ (Q_0)$

$$\begin{aligned}
&\textbf{Eval}\ [\![\texttt{abs}\ (Q_0)\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ \textbf{Eval}\ [\![\texttt{abs}\ (Q_0\uparrow\tbinom{k}{j})]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad=\ |\textbf{Eval}\ [\![Q_0\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)|\\
&\quad=\ |\textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0))|\ \ (\text{by induction})\\
&\quad=\ \textbf{Eval}\ [\![\texttt{abs}\ (Q_0)]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q \equiv Q_0 + Q_1$ (similarly for -, *, /, `mod`)

$$\textbf{Eval}\ [\![(Q_0\ +\ Q_1)\uparrow\tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)$$

$$= \textbf{Eval}\ \left[\!\left[(Q_0 \uparrow \tbinom{k}{j}) \ + \ (Q_1 \uparrow \tbinom{k}{j})\right]\!\right] (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)$$

$$= \begin{array}{l} \textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) + \\ \textbf{Eval}\ [\![Q_1 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \end{array}$$

$$= \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \ + \ \textbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \quad \text{(by induction)}$$

$$= \textbf{Eval}\ [\![Q_0 \ + \ Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)$$

Case: $Q \ \equiv \ Q_0$ `intersect` Q_1 (similarly for `union`, `except`)

$$\textbf{Eval}\ \left[\!\left[(Q_0\ \texttt{intersect}\ Q_1) \uparrow \tbinom{k}{j}\right]\!\right] (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)$$

$$= \textbf{Eval}\ \left[\!\left[(Q_0 \uparrow \tbinom{k}{j})\ \texttt{intersect}\ (Q_1 \uparrow \tbinom{k}{j})\right]\!\right] (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)$$

$$= \begin{array}{l} \textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \cap \\ \textbf{Eval}\ [\![Q_1 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \end{array}$$

$$= \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \cap \textbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \quad \text{(by induction)}$$

$$= \textbf{Eval}\ [\![Q_0\ \texttt{intersect}\ Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)$$

Case: $Q \ \equiv$ `set` $(Q_0, \ldots,\ Q_m)$

$$\textbf{Eval}\ \left[\!\left[(\texttt{set}\ (Q_0, \ldots,\ Q_m)) \uparrow \tbinom{k}{j}\right]\!\right] (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)$$

$$= \textbf{Eval}\ \left[\!\left[\texttt{set}\ (Q_0 \uparrow \tbinom{k}{j}), \ldots,\ Q_m \uparrow \tbinom{k}{j})\right]\!\right] (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)$$

$$= \begin{array}{l} \{\textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0), \ldots, \\ \textbf{Eval}\ [\![Q_m \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)\} \end{array}$$

$$= \{\textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0), \ldots,\ \textbf{Eval}\ [\![Q_m]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)\} \quad \text{(by induction)}$$

$$= \textbf{Eval}\ [\![\texttt{set}\ (Q_0, \ldots,\ Q_m)]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)$$

Case: $Q \ \equiv$ `element` (Q_0)

(if **Eval** $[\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \ = \ \{x\}$)

$$\begin{aligned} & \textbf{Eval}\ \left[\!\left[\texttt{element}\ (Q_0) \uparrow \tbinom{k}{j}\right]\!\right] (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\ &\quad = \ \textbf{Eval}\ \left[\!\left[\texttt{element}\ (Q_0 \uparrow \tbinom{k}{j})\right]\!\right] (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\ &\quad = \ x, \ \ (\text{if } \textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \ = \ \{x\}) \\ &\quad = \ x, \ \ (\text{if } \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \ = \ \{x\}) \ \ \text{(by induction)} \\ &\quad = \ \textbf{Eval}\ [\![\texttt{element}\ (Q_0)]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \end{aligned}$$

Case: $Q \ \equiv$ `flatten` (Q_0)

$$\begin{aligned} & \textbf{Eval}\ \left[\!\left[\texttt{flatten}\ (Q_0) \uparrow \tbinom{k}{j}\right]\!\right] (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\ &\quad = \ \textbf{Eval}\ \left[\!\left[\texttt{flatten}\ (Q_0 \uparrow \tbinom{k}{j})\right]\!\right] (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\ &\quad = \ \{y \mid \exists\, x \ \in \ \textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)\ (y \ \in \ x)\} \\ &\quad = \ \{y \mid \exists\, x \ \in \ \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)\ (y \ \in \ x)\} \ \ \text{(by induction)} \\ &\quad = \ \textbf{Eval}\ [\![\texttt{flatten}\ (Q_0)]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \end{aligned}$$

Case: $Q \equiv \texttt{count}\ (Q_0)$ (similarly for $\texttt{sum}$ and $\texttt{avg}$)

$$\begin{aligned}
&\textbf{Eval}\ [\![\texttt{count}\ (Q_0) \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad = \textbf{Eval}\ [\![\texttt{count}\ (Q_0 \uparrow \tbinom{k}{j})]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad = |\textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)|\\
&\quad = |\textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)|\ \ \text{(by induction)}\\
&\quad = \textbf{Eval}\ [\![\texttt{count}\ (Q_0)]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q \equiv \texttt{min}\ (Q_0)$ (similarly for $\texttt{max}$)

$$\begin{aligned}
&\textbf{Eval}\ [\![\texttt{min}\ (Q_0) \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad = \textbf{Eval}\ [\![\texttt{min}\ (Q_0 \uparrow \tbinom{k}{j})]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad = x,\ \begin{array}{l}(\text{if } x\ \in\ \textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0) \text{ and}\\ \quad \forall\ y\ \in\ \textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\ (x\ \leq\ y))\end{array}\\
&\quad = x,\ \begin{array}{l}(\text{if } x\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0) \text{ and}\\ \quad \forall\ y\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)\ (x\ \leq\ y))\end{array}\quad \text{(by induction)}\\
&\quad = \textbf{Eval}\ [\![\texttt{min}\ (Q_0)]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q \equiv \texttt{select distinct}\ Q_s\ \texttt{from}\ Q_0,\ldots,\ Q_m\ \texttt{where}\ Q_w$

$$\begin{aligned}
&\textbf{Eval}\ \left[\!\!\left[\left(\begin{array}{l}\texttt{select distinct}\ Q_s\\ \texttt{from}\ Q_0,\ldots,\ Q_m\\ \texttt{where}\ Q_w\end{array}\right) \uparrow \tbinom{k}{j}\right]\!\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad = \textbf{Eval}\ \left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ Q_s \uparrow \tbinom{k}{j+m+1}\\ \texttt{from}\ Q_0 \uparrow \tbinom{k}{j},\ldots,\ Q_m \uparrow \tbinom{k}{j+m}\\ \texttt{where}\ Q_w \uparrow \tbinom{k}{j+m+1}\end{array}\right]\!\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad = \left\{\begin{array}{l}\textbf{Eval}\ [\![Q_s \uparrow \tbinom{k}{j+m+1}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0,\ y_0,\ldots,\ y_m)\ |\\ \ y_0\ \in\ \textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0),\ldots,\\ \ y_m\ \in\ \textbf{Eval}\ [\![Q_m \uparrow \tbinom{k}{j+m}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0,\ y_0,\ldots,\ y_{m-1}),\\ \textbf{Eval}\ [\![Q_w \uparrow \tbinom{k}{j+m+1}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0,\ y_0,\ldots,\ y_m)\ =\ \texttt{true}\end{array}\right\}\\
&\quad = \left\{\begin{array}{l}\textbf{Eval}\ [\![Q_s]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0,\ y_0,\ldots,\ y_m)\ |\\ \ y_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0),\ldots,\\ \ y_m\ \in\ \textbf{Eval}\ [\![Q_m]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0,\ y_0,\ldots,\ y_{m-1}),\\ \textbf{Eval}\ [\![Q_w]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0,\ y_0,\ldots,\ y_m)\ =\ \texttt{true}\end{array}\right\}\ \text{(by induction)}\\
&\quad = \textbf{Eval}\ \left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ Q_s\\ \texttt{from}\ Q_0,\ldots,\ Q_m\\ \texttt{where}\ Q_w\end{array}\right]\!\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q \equiv \texttt{select distinct}\ Q_s\ \texttt{from}\ Q_0,\ldots,\ Q_m\ \texttt{where}\ Q_w\ \texttt{groupby}\ Q_{m+1},\ldots,\ Q_p$

$$\begin{aligned}
&\textbf{Eval}\ \left[\!\!\left[\left(\begin{array}{l}\texttt{select distinct}\ Q_s\\ \texttt{from}\ Q_0,\ldots,\ Q_m\\ \texttt{where}\ Q_w\\ \texttt{groupby}\ Q_{m+1},\ldots,\ Q_p\end{array}\right) \uparrow \tbinom{k}{j}\right]\!\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&\quad = \textbf{Eval}\ \left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ Q_s \uparrow \tbinom{k}{j+m+1}\\ \texttt{from}\ Q_0 \uparrow \tbinom{k}{j},\ldots,\ Q_m \uparrow \tbinom{k}{j+m}\\ \texttt{where}\ Q_w \uparrow \tbinom{k}{j+m+1}\\ \texttt{groupby}\ Q_{m+1} \uparrow \tbinom{k}{j+1},\ldots,\ Q_p \uparrow \tbinom{k}{j+1}\end{array}\right]\!\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

$$
= \left\{ \begin{array}{l}
(\epsilon,\ e_{m+1},\ldots,e_p, f\ (x)) \mid\ x\ \in\ R, \\
\quad e_{m+1}\ =\ \begin{cases} 1, & \text{if } \mathbf{Eval}\ \left[\!\left[Q_{m+1} \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \left[\!\left[Q_{m+1} \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{false} \end{cases} ,\ldots, \\
\quad e_p\ =\ \begin{cases} 1, & \text{if } \mathbf{Eval}\ \left[\!\left[Q_p \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \left[\!\left[Q_p \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{false} \end{cases} \\
\quad f(x)\ =\ \left\{ \begin{array}{l}
y\ \in\ R \mid \\
\quad \mathbf{Eval}\ \left[\!\left[Q_{m+1} \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ = \\
\qquad \mathbf{Eval}\ \left[\!\left[Q_{m+1} \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, y), \\
\quad \ldots, \\
\quad \mathbf{Eval}\ \left[\!\left[Q_p \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ = \\
\qquad \mathbf{Eval}\ \left[\!\left[Q_p \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, y)
\end{array} \right\}
\end{array} \right\}
$$

$$
(\text{where } R = \mathbf{Eval}\ \left[\!\!\left[\begin{array}{l} \texttt{select distinct } Q_s \uparrow \binom{k}{j+m+1} \\ \texttt{from } Q_0 \uparrow \binom{k}{j},\ldots,\ Q_m \uparrow \binom{k}{j+m} \\ \texttt{where } Q_w \uparrow \binom{k}{j+m+1} \end{array} \right]\!\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0))
$$

$$
= \left\{ \begin{array}{l}
(\epsilon,\ e_{m+1},\ldots,e_p, f\ (x)) \mid\ x\ \in\ R, \\
\quad e_{m+1}\ =\ \begin{cases} 1, & \text{if } \mathbf{Eval}\ \left[\!\left[Q_{m+1} \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \left[\!\left[Q_{m+1} \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{false} \end{cases} ,\ldots, \\
\quad e_p\ =\ \begin{cases} 1, & \text{if } \mathbf{Eval}\ \left[\!\left[Q_p \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \left[\!\left[Q_p \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{false} \end{cases} \\
\quad f(x)\ =\ \left\{ \begin{array}{l}
y\ \in\ R \mid \\
\quad \mathbf{Eval}\ \left[\!\left[Q_{m+1} \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ = \\
\qquad \mathbf{Eval}\ \left[\!\left[Q_{m+1} \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, y), \\
\quad \ldots, \\
\quad \mathbf{Eval}\ \left[\!\left[Q_p \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ = \\
\qquad \mathbf{Eval}\ \left[\!\left[Q_p \uparrow \binom{k}{j+1}\right]\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, y)
\end{array} \right\}
\end{array} \right\}
$$

$$
(\text{where } R = \mathbf{Eval}\ \left[\!\!\left[\left(\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0,\ldots,\ Q_m \\ \texttt{where } Q_w \end{array} \right) \uparrow \binom{k}{j} \right]\!\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0))
$$

$$
= \left\{ \begin{array}{l}
(\epsilon,\ e_{m+1},\ldots,e_p, f\ (x)) \mid\ x\ \in\ R, \\
\quad e_{m+1}\ =\ \begin{cases} 1, & \text{if } \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{false} \end{cases} ,\ldots, \\
\quad e_p\ =\ \begin{cases} 1, & \text{if } \mathbf{Eval}\ [\![Q_p]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ [\![Q_p]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0, x)\ =\ \mathit{false} \end{cases} \\
\quad f(x)\ =\ \left\{ \begin{array}{l}
y\ \in\ R \mid \\
\quad \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0, x)\ = \\
\qquad \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0, y), \\
\quad \ldots, \\
\quad \mathbf{Eval}\ [\![Q_p]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0, x)\ = \\
\qquad \mathbf{Eval}\ [\![Q_p]\!]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0, y)
\end{array} \right\}
\end{array} \right\}
$$

$$
(\text{where } R = \mathbf{Eval}\ \left[\!\!\left[\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0,\ldots,\ Q_m \\ \texttt{where } Q_w \end{array} \right]\!\!\right]\ (\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0))
$$

(by induction)

$$= \quad \textbf{Eval} \left[\!\!\left[\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0, \ldots, \ Q_m \\ \texttt{where } Q_w \\ \texttt{groupby } Q_{m+1}, \ldots, \ Q_p \end{array} \right]\!\!\right] (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0)$$

Case: $Q \ \equiv \ \texttt{if } Q_0 \texttt{ then } Q_1 \texttt{ else } Q_2$

$$\begin{array}{rl} \textbf{Eval} & [\![(\texttt{if } Q_0 \texttt{ then } Q_1 \texttt{ else } Q_2) \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \\ = & \textbf{Eval} \ \ [\![\texttt{if } (Q_0 \uparrow \binom{k}{j}) \texttt{ then } (Q_1 \uparrow \binom{k}{j}) \texttt{ else } (Q_2 \uparrow \binom{k}{j})]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \\ = & \left\{ \begin{array}{l} \textbf{Eval} \ \ [\![Q_1 \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0), \\ \quad \text{if } \textbf{Eval} \ \ [\![Q_0 \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \ = \ \mathit{true}, \\ \textbf{Eval} \ \ [\![Q_2 \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0), \\ \quad \text{if } \textbf{Eval} \ \ [\![Q_0 \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \ = \ \mathit{false} \end{array} \right. \\ = & \left\{ \begin{array}{l} \textbf{Eval} \ \ [\![Q_1]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0), \\ \quad \text{if } \textbf{Eval} \ \ [\![Q_0]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0) \ = \ \mathit{true}, \\ \textbf{Eval} \ \ [\![Q_2]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0), \\ \quad \text{if } \textbf{Eval} \ \ [\![Q_0]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0) \ = \ \mathit{false}, \end{array} \right. \quad \text{(by induction)} \\ = & \textbf{Eval} \ \ [\![\texttt{if } Q_0 \texttt{ then } Q_1 \texttt{ else } Q_2]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0) \end{array}$$

Case: $Q \ \equiv \ (Q_0)$

$$\begin{array}{rl} \textbf{Eval} & [\![(Q_0) \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \\ = & (\textbf{Eval} \ \ [\![Q_0 \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0)) \\ = & (\textbf{Eval} \ \ [\![Q_0]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0)) \ \ \text{(by induction)} \\ = & \textbf{Eval} \ \ [\![(Q_0)]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0) \end{array}$$

Case: $Q \ \equiv \ Q_0 \ == \ Q_1$ (and similarly for $!=, <, >, \leq, \geq$)

$$\begin{array}{rl} \textbf{Eval} & [\![(Q_0 \ == \ Q_1) \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \\ = & \textbf{Eval} \ \ [\![(Q_0 \uparrow \binom{k}{j}) \ == \ (Q_1 \uparrow \binom{k}{j})]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \\ = & \begin{array}{l} \textbf{Eval} \ \ [\![Q_0 \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \ = \\ \quad \textbf{Eval} \ \ [\![Q_1 \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \end{array} \\ = & \textbf{Eval} \ \ [\![Q_0]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0) \ = \ \textbf{Eval} \ \ [\![Q_1]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0) \ \ \text{(by induction)} \\ = & \textbf{Eval} \ \ [\![Q_0 \ == \ Q_1]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ x_{j-1}, \ldots, \ x_0) \end{array}$$

Case: $Q \ \equiv \ Q_0 \texttt{ in } Q_1$

$$\begin{array}{rl} \textbf{Eval} & [\![(Q_0 \texttt{ in } Q_1) \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \\ = & \textbf{Eval} \ \ [\![(Q_0 \uparrow \binom{k}{j}) \texttt{ in } (Q_1 \uparrow \binom{k}{j})]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \\ = & \begin{array}{l} \textbf{Eval} \ \ [\![Q_0 \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \ \in \\ \quad \textbf{Eval} \ \ [\![Q_1 \uparrow \binom{k}{j}]\!] \ (\epsilon, \ x_n, \ldots, \ x_j, \ d_k, \ldots, \ d_1, \ x_{j-1}, \ldots, \ x_0) \end{array} \end{array}$$

$$= \textbf{Eval } [\![Q_0]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)\ \in\ \textbf{Eval } [\![Q_1]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)\quad \text{(by induction)}$$
$$= \textbf{Eval } [\![Q_0 \texttt{ in } Q_1]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)$$

Case: $Q\ \equiv\ \texttt{not}\ (Q_0)$

$$\begin{aligned}
&\textbf{Eval } [\![\texttt{not}\ (Q_0) \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&= \textbf{Eval } [\![\texttt{not}\ (Q_0 \uparrow \tbinom{k}{j})]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&= \neg\,(\textbf{Eval } [\![Q_0 \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0))\\
&= \neg\,(\textbf{Eval } [\![Q_0]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0))\quad \text{(by induction)}\\
&= \textbf{Eval } [\![\texttt{not}\ (Q_0)]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q\ \equiv\ Q_0 \texttt{ and } Q_1$ (and similarly for `or`)

$$\begin{aligned}
&\textbf{Eval } [\![(Q_0 \texttt{ and } Q_1) \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&= \textbf{Eval } [\![(Q_0 \uparrow \tbinom{k}{j}) \texttt{ and } (Q_1 \uparrow \tbinom{k}{j})]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&= \begin{array}{c}\textbf{Eval } [\![Q_0 \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\ \wedge\\ \textbf{Eval } [\![Q_1 \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\end{array}\\
&= \textbf{Eval } [\![Q_0]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)\ \wedge\ \textbf{Eval } [\![Q_1]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)\quad \text{(by induction)}\\
&= \textbf{Eval } [\![Q_0 \texttt{ and } Q_1]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q\ \equiv\ \texttt{for\_all}\ Q_0\ :\ Q_1$

$$\begin{aligned}
&\textbf{Eval } [\![(\texttt{for_all}\ Q_0\ :\ Q_1) \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&= \textbf{Eval } [\![\texttt{for_all}\ (Q_0 \uparrow \tbinom{k}{j})\ :\ (Q_1 \uparrow \tbinom{k}{j})]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&= \begin{array}{l}\forall\ x\ \in\ \textbf{Eval } [\![Q_0 \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\ \quad (\textbf{Eval } [\![Q_1 \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\ =\ \mathit{true})\end{array}\\
&= \begin{array}{l}\forall\ x\ \in\ \textbf{Eval } [\![Q_0]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)\\ \quad (\textbf{Eval } [\![Q_1]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)\ =\ \mathit{true})\end{array}\quad \text{(by induction)}\\
&= \textbf{Eval } [\![\texttt{for_all}\ Q_0\ :\ Q_1]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q\ \equiv\ \texttt{exists}\ Q_0\ :\ Q_1$

$$\begin{aligned}
&\textbf{Eval } [\![(\texttt{exists}\ Q_0\ :\ Q_1) \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&= \textbf{Eval } [\![\texttt{exists}\ (Q_0 \uparrow \tbinom{k}{j})\ :\ (Q_1 \uparrow \tbinom{k}{j})]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\
&= \begin{array}{l}\exists\ x\ \in\ \textbf{Eval } [\![Q_0 \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\\ \quad (\textbf{Eval } [\![Q_1 \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)\ =\ \mathit{true})\end{array}\\
&= \begin{array}{l}\exists\ x\ \in\ \textbf{Eval } [\![Q_0]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)\\ \quad (\textbf{Eval } [\![Q_1]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)\ =\ \mathit{true})\end{array}\quad \text{(by induction)}\\
&= \textbf{Eval } [\![\texttt{exists}\ Q_0\ :\ Q_1]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ x_{j-1},\ldots,\ x_0)
\end{aligned}$$

Case: $Q\ \equiv\ \texttt{exists}\ (Q_0)$

$$\textbf{Eval } [\![\texttt{exists}\ (Q_0) \uparrow \tbinom{k}{j}]\!]\,(\epsilon,\ x_n,\ldots,\ x_j,\ d_k,\ldots,\ d_1,\ x_{j-1},\ldots,\ x_0)$$

$$
\begin{aligned}
&= \textbf{Eval } [\![\texttt{exists}\,(Q_0 \uparrow \tbinom{k}{j})]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&= \exists\, x\ \in\ \textbf{Eval } [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&= \exists\, x\ \in\ \textbf{Eval } [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \ \ \text{(by induction)} \\
&= \textbf{Eval } [\![\texttt{exists}\,(Q_0)]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q \equiv \texttt{unique}\,(Q_0)$

$$
\begin{aligned}
&\textbf{Eval } [\![\texttt{unique}\,(Q_0) \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&\quad= \textbf{Eval } [\![\texttt{unique}\,(Q_0 \uparrow \tbinom{k}{j})]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&\quad= |\textbf{Eval } [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)| = 1 \\
&\quad= |\textbf{Eval } [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)| = 1 \ \ \text{(by induction)} \\
&\quad= \textbf{Eval } [\![\texttt{unique}\,(Q_0)]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q \equiv Q_0\ \texttt{==}\ \texttt{some}\,(Q_1)$ (and similarly for $!=, <, >, \leq, \geq$)

$$
\begin{aligned}
&\textbf{Eval } [\![(Q_0\ \texttt{==}\ \texttt{some}\,(Q_1)) \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&\quad= \textbf{Eval } [\![(Q_0 \uparrow \tbinom{k}{j})\ \texttt{==}\ \texttt{some}\,(Q_1 \uparrow \tbinom{k}{j})]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&\quad= \begin{array}{l} \exists\, x\ \in\ \textbf{Eval } [\![Q_1 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\ \quad (\textbf{Eval } [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)\ =\ x) \end{array} \\
&\quad= \begin{array}{l} \exists\, x\ \in\ \textbf{Eval } [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \\ \quad (\textbf{Eval } [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)\ =\ x) \end{array} \ \ \text{(by induction)} \\
&\quad= \textbf{Eval } [\![Q_0\ \texttt{==}\ \texttt{some}\,(Q_1)]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q \equiv Q_0\ \texttt{==}\ \texttt{all}\,(Q_1)$ (and similarly for $!=, <, >, \leq, \geq$)

$$
\begin{aligned}
&\textbf{Eval } [\![(Q_0\ \texttt{==}\ \texttt{all}\,(Q_1)) \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&\quad= \textbf{Eval } [\![(Q_0 \uparrow \tbinom{k}{j})\ \texttt{==}\ \texttt{all}\,(Q_1 \uparrow \tbinom{k}{j})]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\
&\quad= \begin{array}{l} \forall\, x\ \in\ \textbf{Eval } [\![Q_1 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0) \\ \quad (\textbf{Eval } [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ d_k, \ldots,\ d_1,\ x_{j-1}, \ldots,\ x_0)\ =\ x) \end{array} \\
&\quad= \begin{array}{l} \forall\, x\ \in\ \textbf{Eval } [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0) \\ \quad (\textbf{Eval } [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)\ =\ x) \end{array} \ \ \text{(by induction)} \\
&\quad= \textbf{Eval } [\![Q_0\ \texttt{==}\ \texttt{all}\,(Q_1)]\!]\ (\epsilon,\ x_n, \ldots,\ x_j,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

□

Lemma 3.2 *Given environment $\rho = (\epsilon,\ x_n, \ldots,\ x_{j+1},\ x_{j-1}, \ldots,\ x_0)$ (each $x_i \in \mathcal{D}$), OQL_{deB} expression Q' that is well-formed wrt ρ and Q that is well-formed wrt $(\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval } [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)$,*

$$\textbf{Eval } [\![Q \downarrow \tbinom{Q'}{j}]\!]\ \rho\ =\ \textbf{Eval } [\![Q]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval } [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0).$$

Proof: (By structural induction on Q) For the base cases we have:

Case: $Q \equiv c$ (for constant or global name, c).

$$
\begin{aligned}
\textbf{Eval } [\![c \downarrow \tbinom{Q'}{j}]\!]\ \rho\ &=\ \textbf{Eval } [\![c]\!]\ \rho \\
&=\ \textbf{Val } [\![c]\!] \\
&=\ \textbf{Eval } [\![c]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval } [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q \;\equiv\; V_i$ (for $0 \le i \le n$)

$$
\begin{aligned}
&\textbf{Eval}\;\; [\![V_i \downarrow \tbinom{Q'}{j}]\!]\;\rho \\
&= \begin{cases} \textbf{Eval}\;\; [\![V_{i-1}]\!]\;\rho, & \text{if } i > j, \\ \textbf{Eval}\;\; [\![Q']\!]\;\rho, & \text{if } i = j, \\ \textbf{Eval}\;\; [\![V_i]\!]\;\rho, & \text{if } i < j \end{cases} \\
&= \begin{cases} \textbf{Eval}\;\; [\![V_{i-1}]\!]\;(\epsilon,\; x_n, \ldots,\; x_i, \ldots,\; x_{j+1},\; x_{j-1}, \ldots,\; x_0), & \text{if } i > j, \\ \textbf{Eval}\;\; [\![Q']\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; x_{j-1}, \ldots,\; x_0), & \text{if } i = j, \\ \textbf{Eval}\;\; [\![V_i]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; x_{j-1}, \ldots,\; x_i, \ldots,\; x_0), & \text{if } i < j \end{cases} \\
&= \begin{cases} \textbf{Eval}\;\; [\![V_i]\!]\;(\epsilon,\; x_n, \ldots x_i, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0), & \text{if } i > j, \\ \textbf{Eval}\;\; [\![Q']\!]\;\rho, & \text{if } i = j, \\ \textbf{Eval}\;\; [\![V_i]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_i, \ldots,\; x_0), & \text{if } i < j \end{cases} \\
&= \textbf{Eval}\;\; [\![V_i]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0)
\end{aligned}
$$

For the inductive cases, we have:

Case: $Q \;\equiv\; \ll Q_0, \ldots,\; Q_m \gg$

$$
\begin{aligned}
&\textbf{Eval}\;\; [\![\ll Q_0, \ldots,\; Q_m \gg \downarrow \tbinom{Q'}{j}]\!]\;\rho \\
&= \textbf{Eval}\;\; [\![\ll Q_0 \downarrow \tbinom{Q'}{j}, \ldots,\; Q_m \downarrow \tbinom{Q'}{j} \gg]\!]\;\rho \\
&= (\textbf{Eval}\;\; [\![Q_0 \downarrow \tbinom{Q'}{j}]\!]\;\rho, \ldots,\; \textbf{Eval}\;\; [\![Q_m \downarrow \tbinom{Q'}{j}]\!]\;\rho) \\
&= \begin{array}{l} (\textbf{Eval}\;\; [\![Q_0]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0), \ldots, \\ \;\; \textbf{Eval}\;\; [\![Q_m]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0)) \end{array} \quad \text{(by induction)} \\
&= \textbf{Eval}\;\; [\![\ll Q_0, \ldots,\; Q_m \gg]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0)
\end{aligned}
$$

Case: $Q \;\equiv\; Q_0\;.\;a_0$ (if $\textbf{Eval}\;\; [\![Q_0]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0) \;=\; (\delta,\; v)$)

$$
\begin{aligned}
&\textbf{Eval}\;\; [\![(Q_0\;.\;a_0) \downarrow \tbinom{Q'}{j}]\!]\;\rho \\
&= \textbf{Eval}\;\; [\![(Q_0 \downarrow \tbinom{Q'}{j})\;.\;a_0]\!]\;\rho \\
&= \delta, \quad (\text{if } \textbf{Eval}\;\; [\![Q_0 \downarrow \tbinom{Q'}{j}]\!]\;\rho \;=\; (\delta,\; v)) \\
&= \delta, \quad (\text{if } \textbf{Eval}\;\; [\![Q_0]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0) \;=\; (\delta,\; v)) \;\; \text{(by induction)} \\
&= \textbf{Eval}\;\; [\![Q_0\;.\;a_0]\!]\;\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0)
\end{aligned}
$$

Case: $Q \;\equiv\; Q_0\;.\;a_1$ (if $\textbf{Eval}\;\; [\![Q_0]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0) \;=\; (\delta,\; v)$)

$$
\begin{aligned}
&\textbf{Eval}\;\; [\![(Q_0\;.\;a_1) \downarrow \tbinom{Q'}{j}]\!]\;\rho \\
&= \textbf{Eval}\;\; [\![(Q_0 \downarrow \tbinom{Q'}{j})\;.\;a_1]\!]\;\rho \\
&= v, \quad (\text{if } \textbf{Eval}\;\; [\![Q_0 \downarrow \tbinom{Q'}{j}]\!]\;\rho \;=\; (\delta,\; v)) \\
&= v, \quad (\text{if } \textbf{Eval}\;\; [\![Q_0]\!]\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0) \;=\; (\delta,\; v)) \;\; \text{(by induction)} \\
&= \textbf{Eval}\;\; [\![Q_0\;.\;a_1]\!]\;\;(\epsilon,\; x_n, \ldots,\; x_{j+1},\; \textbf{Eval}\;\; [\![Q']\!]\;\rho,\; x_{j-1}, \ldots,\; x_0)
\end{aligned}
$$

Case: $Q \;\equiv\; Q_0 \rightarrow \texttt{att}$ (if *att* is defined for the type of Q_0)

$$
\textbf{Eval}\;\; [\![(Q_0 \rightarrow\; \texttt{att}) \downarrow \tbinom{Q'}{j}]\!]\;\rho
$$

$$
\begin{aligned}
&= \textbf{Eval}\ \llbracket (Q_0 \downarrow \tbinom{Q'}{j}) \rightarrow\ \texttt{att} \rrbracket\ \rho \\
&= \mathit{att}\ (\textbf{Eval}\ \llbracket Q_0 \downarrow \tbinom{Q'}{j} \rrbracket\ \rho) \\
&= \mathit{att}\ (\textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0) \quad \text{(by induction)} \\
&= \textbf{Eval}\ \llbracket Q_0 \rightarrow\ \texttt{att} \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q\ \equiv\ -\ Q_0$

$$
\begin{aligned}
&\textbf{Eval}\ \llbracket (-\ Q_0) \downarrow \tbinom{Q'}{j} \rrbracket\ \rho \\
&\quad = \textbf{Eval}\ \llbracket -(Q_0 \downarrow \tbinom{Q'}{j}) \rrbracket\ \rho \\
&\quad = -\ (\textbf{Eval}\ \llbracket Q_0 \downarrow \tbinom{Q'}{j} \rrbracket\ \rho) \\
&\quad = -\ (\textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)) \quad \text{(by induction)} \\
&\quad = \textbf{Eval}\ \llbracket -\ Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q\ \equiv\ \texttt{abs}\ (Q_0)$

$$
\begin{aligned}
&\textbf{Eval}\ \llbracket \texttt{abs}\ (Q_0) \downarrow \tbinom{Q'}{j} \rrbracket\ \rho \\
&\quad = \textbf{Eval}\ \llbracket \texttt{abs}\ (Q_0 \downarrow \tbinom{Q'}{j}) \rrbracket\ \rho \\
&\quad = |\textbf{Eval}\ \llbracket Q_0 \downarrow \tbinom{Q'}{j} \rrbracket\ \rho| \\
&\quad = |\textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0))| \quad \text{(by induction)} \\
&\quad = \textbf{Eval}\ \llbracket \texttt{abs}\ (Q_0) \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q\ \equiv\ Q_0\ +\ Q_1$ (similarly for -, *, /, `mod`)

$$
\begin{aligned}
&\textbf{Eval}\ \llbracket (Q_0\ +\ Q_1) \downarrow \tbinom{Q'}{j} \rrbracket\ (\rho) \\
&\quad = \textbf{Eval}\ \llbracket (Q_0 \downarrow \tbinom{Q'}{j})\ +\ (Q_1 \downarrow \tbinom{Q'}{j}) \rrbracket\ \rho \\
&\quad = \textbf{Eval}\ \llbracket Q_0 \downarrow \tbinom{Q'}{j} \rrbracket\ \rho\ +\ \textbf{Eval}\ \llbracket Q_1 \downarrow \tbinom{Q'}{j} \rrbracket\ \rho \\
&\quad = \begin{array}{c} \textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)\ + \\ \textbf{Eval}\ \llbracket Q_1 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0) \end{array} \quad \text{(by induction)} \\
&\quad = \textbf{Eval}\ \llbracket Q_0\ +\ Q_1 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q\ \equiv\ Q_0\ \texttt{intersect}\ Q_1$ (similarly for `union`, `except`)

$$
\begin{aligned}
&\textbf{Eval}\ \llbracket (Q_0\ \texttt{intersect}\ Q_1) \downarrow \tbinom{Q'}{j} \rrbracket\ \rho \\
&\quad = \textbf{Eval}\ \llbracket (Q_0 \downarrow \tbinom{Q'}{j})\ \texttt{intersect}\ (Q_1 \downarrow \tbinom{Q'}{j}) \rrbracket\ \rho \\
&\quad = \textbf{Eval}\ \llbracket Q_0 \downarrow \tbinom{Q'}{j} \rrbracket\ \rho\ \cap\ \textbf{Eval}\ \llbracket Q_1 \downarrow \tbinom{Q'}{j} \rrbracket\ \rho \\
&\quad = \begin{array}{c} \textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)\ \cap \\ \textbf{Eval}\ \llbracket Q_1 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0) \end{array} \quad \text{(by induction)} \\
&\quad = \textbf{Eval}\ \llbracket Q_0\ \texttt{intersect}\ Q_1 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q\ \equiv\ \texttt{set}\ (Q_0, \ldots,\ Q_m)$

$$
\begin{aligned}
&\textbf{Eval}\ \llbracket \texttt{set}\ (Q_0, \ldots,\ Q_m) \downarrow \tbinom{Q'}{j} \rrbracket\ \rho \\
&\quad = \textbf{Eval}\ \llbracket \texttt{set}\ (Q_0 \downarrow \tbinom{Q'}{j}, \ldots,\ Q_m \downarrow \tbinom{Q'}{j}) \rrbracket\ \rho
\end{aligned}
$$

$$
\begin{array}{ll}
= & \{\mathbf{Eval}\ \llbracket Q_0 \downarrow \binom{Q'}{j} \rrbracket\ \rho, \ldots,\ \mathbf{Eval}\ \llbracket Q_n \downarrow \binom{Q'}{j} \rrbracket\ \rho\} \\
= & \begin{array}{l} \{\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0), \ldots, \\ \quad \mathbf{Eval}\ \llbracket Q_1 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)\} \end{array} \quad \text{(by induction)} \\
= & \mathbf{Eval}\ \llbracket \texttt{set}\ (Q_0, \ldots,\ Q_n) \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{array}
$$

Case: $Q \equiv \texttt{element}\ (Q_0)$
(if $\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0) = \{x\}$)

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathbf{Eval}\ \llbracket \texttt{element}\ (Q_0) \downarrow \binom{Q'}{j} \rrbracket\ \rho} \\
= & \mathbf{Eval}\ \llbracket \texttt{element}\ (Q_0 \downarrow \binom{Q'}{j}) \rrbracket\ \rho \\
= & x,\ \ (\text{if } \mathbf{Eval}\ \llbracket Q_0 \downarrow \binom{Q'}{j} \rrbracket\ \rho = \{x\}) \\
= & x,\ \ (\text{if } \mathbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0) = \{x\}) \quad \text{(by induction)} \\
= & \mathbf{Eval}\ \llbracket \texttt{element}\ (Q_0) \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{array}
$$

Case: $Q \equiv \texttt{flatten}\ (Q_0)$

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathbf{Eval}\ \llbracket \texttt{flatten}\ (Q_0) \downarrow \binom{Q'}{j} \rrbracket\ \rho} \\
= & \mathbf{Eval}\ \llbracket \texttt{flatten}\ (Q_0 \downarrow \binom{Q'}{j}) \rrbracket\ \rho \\
= & \{y \mid \exists\, x \in \mathbf{Eval}\ \llbracket Q_0 \downarrow \binom{Q'}{j} \rrbracket\ \rho\ (y \in x)\} \\
= & \{y \mid \exists\, x \in \mathbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)\ (y \in x)\} \quad \text{(by induction)} \\
= & \mathbf{Eval}\ \llbracket \texttt{flatten}\ (Q_0) \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{array}
$$

Case: $Q \equiv \texttt{count}\ (Q_0)$ (similarly for `sum` and `avg`)

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathbf{Eval}\ \llbracket \texttt{count}\ (Q_0) \downarrow \binom{Q'}{j} \rrbracket\ \rho} \\
= & \mathbf{Eval}\ \llbracket \texttt{count}\ (Q_0 \downarrow \binom{Q'}{j}) \rrbracket\ \rho \\
= & |\mathbf{Eval}\ \llbracket Q_0 \downarrow \binom{Q'}{j} \rrbracket\ \rho| \\
= & |\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)| \\
= & \mathbf{Eval}\ \llbracket \texttt{count}\ (Q_0) \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{array}
$$

Case: $Q \equiv \texttt{min}\ (Q_0)$ (similarly for `max`)

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathbf{Eval}\ \llbracket \texttt{min}\ (Q_0) \downarrow \binom{Q'}{j} \rrbracket\ \rho} \\
= & \mathbf{Eval}\ \llbracket \texttt{min}\ (Q_0 \downarrow \binom{Q'}{j}) \rrbracket\ \rho \\
= & x, (\text{if } x \in \mathbf{Eval}\ \llbracket Q_0 \downarrow \binom{Q'}{j} \rrbracket\ \rho \text{ and } \forall\, y \in \mathbf{Eval}\ \llbracket Q_0 \downarrow \binom{Q'}{j} \rrbracket\ \rho\ (x \leq y)) \\
= & x,\ \begin{array}{l} (\text{if } x \in \mathbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0) \text{ and} \\ \ \forall\, y \in \mathbf{Eval}\ \llbracket Q_0 \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)\ (x \leq y)) \end{array} \quad \text{(by induction)} \\
= & \mathbf{Eval}\ \llbracket \texttt{min}\ (Q_0) \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{array}
$$

Case: $Q \equiv \texttt{select distinct}\ Q_s\ \texttt{from}\ Q_0, \ldots,\ Q_m\ \texttt{where}\ Q_w$

$$
\mathbf{Eval}\ \left\llbracket \left(\begin{array}{l} \texttt{select distinct}\ Q_s \\ \texttt{from}\ Q_0, \ldots,\ Q_m \\ \texttt{where}\ Q_w \end{array} \right) \downarrow \binom{Q'}{j} \right\rrbracket\ \rho
$$

$$= \quad \mathbf{Eval} \left\llbracket \begin{array}{l} \texttt{select distinct } Q_s \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1} \\ \texttt{from } Q_0 \downarrow \binom{Q'}{j}, \ldots,\ Q_m \downarrow \binom{Q' \uparrow \binom{m}{0}}{j+m} \\ \texttt{where } Q_w \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1} \end{array} \right\rrbracket \rho$$

$$= \left\{ \begin{array}{l} \mathbf{Eval} \ \left\llbracket Q_s \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1} \right\rrbracket (\rho,\ y_0, \ldots,\ y_m) \mid \\ \ y_0 \in \mathbf{Eval} \ \left\llbracket Q_0 \downarrow \binom{Q'}{j} \right\rrbracket \rho, \ldots, \\ \ y_m \in \mathbf{Eval} \ \left\llbracket Q_m \downarrow \binom{Q' \uparrow \binom{m}{0}}{j+m} \right\rrbracket (\rho,\ y_0, \ldots,\ y_{m-1}), \\ \mathbf{Eval} \ \left\llbracket Q_w \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1} \right\rrbracket (\rho,\ y_0, \ldots,\ y_m) = \texttt{true} \end{array} \right\}$$

$$= \left\{ \begin{array}{l} \mathbf{Eval} \ \llbracket Q_s \rrbracket (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval} \ \llbracket Q' \uparrow \binom{m+1}{0} \rrbracket (\rho,\ y_0, \ldots,\ y_m),\ x_{j-1}, \ldots,\ x_0,\ y_0, \ldots,\ y_m) \mid \\ \ y_0 \in \mathbf{Eval} \ \llbracket Q_0 \rrbracket (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval} \ \llbracket Q' \rrbracket \rho,\ x_{j-1}, \ldots,\ x_0), \ldots, \\ \ y_m \in \mathbf{Eval} \ \llbracket Q_m \rrbracket (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval} \ \llbracket Q' \uparrow \binom{m}{0} \rrbracket (\rho,\ y_0, \ldots,\ y_{m-1}),\ x_{j-1}, \ldots,\ x_0,\ y_0, \ldots,\ y_{m-1}), \\ \mathbf{Eval} \ \llbracket Q_w \rrbracket (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval} \ \llbracket Q' \uparrow \binom{m+1}{0} \rrbracket (\rho,\ y_0, \ldots,\ y_m),\ x_{j-1}, \ldots,\ x_0,\ y_0, \ldots,\ y_m) = \texttt{true} \end{array} \right\}$$

(by induction)

$$= \left\{ \begin{array}{l} \mathbf{Eval} \ \llbracket Q_s \rrbracket (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval} \ \llbracket Q' \rrbracket \rho,\ x_{j-1}, \ldots,\ x_0,\ y_0, \ldots,\ y_m) \mid \\ \ y_0 \in \mathbf{Eval} \ \llbracket Q_0 \rrbracket (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval} \ \llbracket Q' \rrbracket \rho,\ x_{j-1}, \ldots,\ x_0), \ldots, \\ \ y_m \in \mathbf{Eval} \ \llbracket Q_m \rrbracket (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval} \ \llbracket Q' \rrbracket \rho,\ x_{j-1}, \ldots,\ x_0,\ y_0, \ldots,\ y_{m-1}), \\ \mathbf{Eval} \ \llbracket Q_w \rrbracket (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval} \ \llbracket Q' \rrbracket \rho,\ x_{j-1}, \ldots,\ x_0,\ y_0, \ldots,\ y_m) = \texttt{true} \end{array} \right\}$$

(by Lemma 3.1)

$$= \quad \mathbf{Eval} \left\llbracket \begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0, \ldots,\ Q_m \\ \texttt{where } Q_w \end{array} \right\rrbracket (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval} \ \llbracket Q' \rrbracket \rho,\ x_{j-1}, \ldots,\ x_0)$$

Case: $Q \equiv \texttt{select distinct } Q_s \texttt{ from } Q_0, \ldots,\ Q_m \texttt{ where } Q_w \texttt{ groupby } Q_{m+1}, \ldots,\ Q_p$

$$\mathbf{Eval} \left\llbracket \left(\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0, \ldots,\ Q_m \\ \texttt{where } Q_w \\ \texttt{groupby } Q_{m+1}, \ldots,\ Q_p \end{array} \right) \downarrow \binom{Q'}{j} \right\rrbracket \rho$$

$$= \quad \mathbf{Eval} \left\llbracket \begin{array}{l} \texttt{select distinct } Q_s \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1} \\ \texttt{from } Q_0 \downarrow \binom{Q'}{j}, \ldots,\ Q_m \downarrow \binom{Q' \uparrow \binom{m}{0}}{j+m} \\ \texttt{where } Q_w \downarrow \binom{Q' \uparrow \binom{m}{0}}{j+m+1} \\ \texttt{groupby } Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1}, \ldots,\ Q_p \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \end{array} \right\rrbracket \rho$$

$$
= \left\{ \begin{array}{l}
(\epsilon,\ e_{m+1}, \ldots, e_p, f\ (x)) \mid\ x\ \in\ R, \\
\quad e_{m+1}\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \mathbf{Eval}\ \left\llbracket Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \left\llbracket Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathit{false} \end{array} \right. , \ldots, \\
\quad e_p\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \mathbf{Eval}\ \left\llbracket Q_p \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \left\llbracket Q_p \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathit{false} \end{array} \right. \\
\quad f(x)\ =\ \left\{ \begin{array}{l} y\ \in\ R\ \mid \\ \quad \mathbf{Eval}\ \left\llbracket Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathbf{Eval}\ \left\llbracket Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, y), \\ \quad \ldots, \\ \quad \mathbf{Eval}\ \left\llbracket Q_p \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathbf{Eval}\ \left\llbracket Q_p \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, y) \end{array} \right\}
\end{array} \right\}
$$

$$
(\text{where } R = \mathbf{Eval}\ \left\llbracket \begin{array}{l} \texttt{select distinct}\ Q_s \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1} \\ \texttt{from}\ Q_0 \downarrow \binom{Q'}{j}, \ldots,\ Q_m \downarrow \binom{Q' \uparrow \binom{m}{0}}{j+m} \\ \texttt{where}\ Q_w \downarrow \binom{Q' \uparrow \binom{m}{0}}{j+m+1} \end{array} \right\rrbracket\ \rho)
$$

$$
= \left\{ \begin{array}{l}
(\epsilon,\ e_{m+1}, \ldots, e_p, f\ (x)) \mid\ x\ \in\ R, \\
\quad e_{m+1}\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \mathbf{Eval}\ \left\llbracket Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \left\llbracket Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathit{false} \end{array} \right. , \ldots, \\
\quad e_p\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \mathbf{Eval}\ \left\llbracket Q_p \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \left\llbracket Q_p \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathit{false} \end{array} \right. \\
\quad f(x)\ =\ \left\{ \begin{array}{l} y\ \in\ R\ \mid \\ \quad \mathbf{Eval}\ \left\llbracket Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathbf{Eval}\ \left\llbracket Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, y), \\ \quad \ldots, \\ \quad \mathbf{Eval}\ \left\llbracket Q_p \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, x)\ =\ \mathbf{Eval}\ \left\llbracket Q_p \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1} \right\rrbracket\ (\rho, y) \end{array} \right\}
\end{array} \right\}
$$

$$
(\text{where } R = \mathbf{Eval}\ \left\llbracket \left(\begin{array}{l} \texttt{select distinct}\ Q_s \\ \texttt{from}\ Q_0, \ldots,\ Q_m \\ \texttt{where}\ Q_w \end{array} \right) \downarrow \binom{Q' \uparrow \binom{m}{0}}{j} \right\rrbracket\ \rho)
$$

$$
= \left\{ \begin{array}{l}
(\epsilon,\ e_{m+1}, \ldots, e_p, f\ (x)) \mid\ x\ \in\ R, \\
\quad e_{m+1}\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \mathbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \uparrow \binom{1}{0} \rrbracket\ (\rho,\ x),\ x_{j-1}, \ldots,\ x_0,\ x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \uparrow \binom{1}{0} \rrbracket\ (\rho,\ x),\ x_{j-1}, \ldots,\ x_0,\ x)\ =\ \mathit{false} \end{array} \right. , \ldots, \\
\quad e_p\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \mathbf{Eval}\ \llbracket Q_p \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \uparrow \binom{1}{0} \rrbracket\ (\rho,\ x),\ x_{j-1}, \ldots,\ x_0,\ x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ \llbracket Q_p \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \uparrow \binom{1}{0} \rrbracket\ (\rho,\ x),\ x_{j-1}, \ldots,\ x_0,\ x)\ =\ \mathit{false} \end{array} \right. \\
\quad f(x)\ =\ \left\{ \begin{array}{l} y\ \in\ R\ \mid \\ \quad \mathbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \uparrow \binom{1}{0} \rrbracket\ (\rho,\ x),\ x_{j-1}, \ldots,\ x_0,\ x)\ = \\ \qquad \mathbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \uparrow \binom{1}{0} \rrbracket\ (\rho,\ x),\ x_{j-1}, \ldots,\ x_0,\ y), \\ \quad \ldots, \\ \quad \mathbf{Eval}\ \llbracket Q_p \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \uparrow \binom{1}{0} \rrbracket\ (\rho,\ x),\ x_{j-1}, \ldots,\ x_0,\ x)\ = \\ \qquad \mathbf{Eval}\ \llbracket Q_p \rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \uparrow \binom{1}{0} \rrbracket\ (\rho,\ x),\ x_{j-1}, \ldots,\ x_0,\ y) \end{array} \right\}
\end{array} \right\}
$$

$$
(\text{where } R = \mathbf{Eval}\ \left\llbracket \begin{array}{l} \texttt{select distinct}\ Q_s \\ \texttt{from}\ Q_0, \ldots,\ Q_m \\ \texttt{where}\ Q_w \end{array} \right\rrbracket\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ \llbracket Q' \uparrow \binom{1}{0} \rrbracket\ (\rho,\ x),\ x_{j-1}, \ldots,\ x_0))
$$

(by induction)

$$= \left\{ \begin{array}{l} (\epsilon,\ e_{m+1}, \ldots, e_p, f\ (x)) \mid\ x\ \in\ R, \\ \quad e_{m+1}\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0,\ x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0,\ x)\ =\ \mathit{false} \end{array} \right. , \ldots, \\ \\ \quad e_p\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \mathbf{Eval}\ [\![Q_p]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0,\ x)\ =\ \mathit{true}, \\ 0, & \text{if } \mathbf{Eval}\ [\![Q_p]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0,\ x)\ =\ \mathit{false} \end{array} \right. \\ \\ \quad f(x)\ =\ \left\{ \begin{array}{l} y\ \in\ R \mid \\ \ \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0,\ x)\ = \\ \quad \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0,\ y), \\ \ \ldots, \\ \ \mathbf{Eval}\ [\![Q_p]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0,\ x)\ = \\ \quad \mathbf{Eval}\ [\![Q_p]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0,\ y) \end{array} \right\} \end{array} \right\}$$

$$(\text{where } R = \mathbf{Eval}\ \left[\!\!\left[\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0, \ldots,\ Q_m \\ \texttt{where } Q_w \end{array} \right]\!\!\right]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0))$$

(By Lemma 3.1)

$$= \mathbf{Eval}\ \left[\!\!\left[\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0, \ldots,\ Q_m \\ \texttt{where } Q_w \\ \texttt{groupby } Q_{m+1}, \ldots,\ Q_p \end{array} \right]\!\!\right]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)$$

Case: $Q\ \equiv\ \texttt{if } Q_0 \texttt{ then } Q_1 \texttt{ else } Q_2$

$$\begin{array}{l} \mathbf{Eval}\ [\![(\texttt{if } Q_0 \texttt{ then } Q_1 \texttt{ else } Q_2) \uparrow \binom{k}{j}]\!]\ \rho \\ \quad =\ \mathbf{Eval}\ [\![\texttt{if } (Q_0 \uparrow \binom{k}{j}) \texttt{ then } (Q_1 \uparrow \binom{k}{j}) \texttt{ else } (Q_2 \uparrow \binom{k}{j})]\!]\ \rho \\ \quad =\ \left\{ \begin{array}{l} \mathbf{Eval}\ [\![Q_1 \uparrow \binom{k}{j}]\!]\ \rho, \text{ if } \mathbf{Eval}\ [\![Q_0 \uparrow \binom{k}{j}]\!]\ \rho\ =\ \mathit{true}, \\ \mathbf{Eval}\ [\![Q_2 \uparrow \binom{k}{j}]\!]\ \rho, \text{ if } \mathbf{Eval}\ [\![Q_0 \uparrow \binom{k}{j}]\!]\ \rho\ =\ \mathit{false} \end{array} \right. \\ \\ \quad =\ \left\{ \begin{array}{l} \mathbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0), \\ \quad \text{if } \mathbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)\ =\ \mathit{true}, \\ \mathbf{Eval}\ [\![Q_2]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0), \\ \quad \text{if } \mathbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)\ =\ \mathit{false}, \end{array} \right. \quad \text{(by induction)} \\ \quad =\ \mathbf{Eval}\ [\![\texttt{if } Q_0 \texttt{ then } Q_1 \texttt{ else } Q_2]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \end{array}$$

Case: $Q\ \equiv\ (Q_0)$

$$\begin{array}{l} \mathbf{Eval}\ [\![(Q_0) \uparrow \binom{k}{j}]\!]\ \rho \\ \quad =\ (\mathbf{Eval}\ [\![Q_0 \uparrow \binom{k}{j}]\!]\ \rho) \\ \quad =\ (\mathbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)) \quad \text{(by induction)} \\ \quad =\ \mathbf{Eval}\ [\![(Q_0)]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \end{array}$$

Case: $Q\ \equiv\ Q_0\ ==\ Q_1$ (and similarly for $!=$, $<$, $>$, $\leq$, $\geq$)

$$\mathbf{Eval}\ [\![(Q_0\ ==\ Q_1) \uparrow \binom{k}{j}]\!]\ \rho$$

$$
\begin{aligned}
&= \ \mathbf{Eval}\ \left[\!\left[(Q_0 \uparrow \tbinom{k}{j}) \ \texttt{==}\ (Q_1 \uparrow \tbinom{k}{j})\right]\!\right] \rho \\
&= \ \mathbf{Eval}\ \left[\!\left[Q_0 \uparrow \tbinom{k}{j}\right]\!\right] \rho \ = \ \mathbf{Eval}\ \left[\!\left[Q_1 \uparrow \tbinom{k}{j}\right]\!\right] \rho \\
&= \ \begin{array}{l} \mathbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \ = \\ \quad \mathbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \end{array} \quad \text{(by induction)} \\
&= \ \mathbf{Eval}\ [\![Q_0 \ \texttt{==}\ Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q \ \equiv \ Q_0 \ \texttt{in}\ Q_1$

$$
\begin{aligned}
&\mathbf{Eval}\ \left[\!\left[(Q_0 \ \texttt{in}\ Q_1) \uparrow \tbinom{k}{j}\right]\!\right] \rho \\
&\quad = \ \mathbf{Eval}\ \left[\!\left[(Q_0 \uparrow \tbinom{k}{j}) \ \texttt{in}\ (Q_1 \uparrow \tbinom{k}{j})\right]\!\right] \rho \\
&\quad = \ \mathbf{Eval}\ \left[\!\left[Q_0 \uparrow \tbinom{k}{j}\right]\!\right] \rho \ \in \ \mathbf{Eval}\ \left[\!\left[Q_1 \uparrow \tbinom{k}{j}\right]\!\right] \rho \\
&\quad = \ \begin{array}{l} \mathbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \ \in \\ \quad \mathbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \end{array} \quad \text{(by induction)} \\
&\quad = \ \mathbf{Eval}\ [\![Q_0 \ \texttt{in}\ Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q \ \equiv \ \texttt{not}\ (Q_0)$

$$
\begin{aligned}
&\mathbf{Eval}\ \left[\!\left[\texttt{not}\ (Q_0) \uparrow \tbinom{k}{j}\right]\!\right] \rho \\
&\quad = \ \mathbf{Eval}\ \left[\!\left[\texttt{not}\ (Q_0 \uparrow \tbinom{k}{j})\right]\!\right] \rho \\
&\quad = \ \neg\, (\mathbf{Eval}\ \left[\!\left[Q_0 \uparrow \tbinom{k}{j}\right]\!\right] \rho) \\
&\quad = \ \neg\, (\mathbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)) \quad \text{(by induction)} \\
&\quad = \ \mathbf{Eval}\ [\![\texttt{not}\ (Q_0)]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q \ \equiv \ Q_0 \ \texttt{and}\ Q_1$ (and similarly for `or`)

$$
\begin{aligned}
&\mathbf{Eval}\ \left[\!\left[(Q_0 \ \texttt{and}\ Q_1) \uparrow \tbinom{k}{j}\right]\!\right] \rho \\
&\quad = \ \mathbf{Eval}\ \left[\!\left[(Q_0 \uparrow \tbinom{k}{j}) \ \texttt{and}\ (Q_1 \uparrow \tbinom{k}{j})\right]\!\right] \rho \\
&\quad = \ \mathbf{Eval}\ \left[\!\left[Q_0 \uparrow \tbinom{k}{j}\right]\!\right] \rho \ \wedge \ \mathbf{Eval}\ \left[\!\left[Q_1 \uparrow \tbinom{k}{j}\right]\!\right] \rho \\
&\quad = \ \begin{array}{l} \mathbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \ \wedge \\ \quad \mathbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \end{array} \quad \text{(by induction)} \\
&\quad = \ \mathbf{Eval}\ [\![Q_0 \ \texttt{and}\ Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q \ \equiv \ \texttt{for\_all}\ Q_0 \ : \ Q_1$

$$
\begin{aligned}
&\mathbf{Eval}\ \left[\!\left[(\texttt{for\_all}\ Q_0 \ : \ Q_1) \uparrow \tbinom{k}{j}\right]\!\right] \rho \\
&\quad = \ \mathbf{Eval}\ \left[\!\left[\texttt{for\_all}\ (Q_0 \uparrow \tbinom{k}{j}) \ : \ (Q_1 \uparrow \tbinom{k}{j})\right]\!\right] \rho \\
&\quad = \ \forall\, x \ \in \ \mathbf{Eval}\ \left[\!\left[Q_0 \uparrow \tbinom{k}{j}\right]\!\right] \rho \ (\mathbf{Eval}\ \left[\!\left[Q_1 \uparrow \tbinom{k}{j}\right]\!\right] \rho \ = \ \mathit{true}) \\
&\quad = \ \begin{array}{l} \forall\, x \ \in \ \mathbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \\ \quad (\mathbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \ = \ \mathit{true}) \end{array} \quad \text{(by induction)} \\
&\quad = \ \mathbf{Eval}\ [\![\texttt{for\_all}\ Q_0 \ : \ Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \mathbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q \ \equiv \ \texttt{exists}\ Q_0 \ : \ Q_1$

$$
\mathbf{Eval}\ \left[\!\left[(\texttt{exists}\ Q_0 \ : \ Q_1) \uparrow \tbinom{k}{j}\right]\!\right] \rho
$$

$$
\begin{aligned}
&= \textbf{Eval}\ [\![\texttt{exists}\,(Q_0 \uparrow \tbinom{k}{j})\ :\ (Q_1 \uparrow \tbinom{k}{j})]\!]\ \rho \\
&= \exists\, x\ \in\ \textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ \rho\ (\textbf{Eval}\ [\![Q_1 \uparrow \tbinom{k}{j}]\!]\ \rho\ =\ \mathit{true}) \\
&= \begin{array}{l}\exists\, x\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \\ \quad (\textbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)\ =\ \mathit{true})\end{array} \quad \text{(by induction)} \\
&= \textbf{Eval}\ [\![\texttt{exists}\, Q_0\ :\ Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q\ \equiv\ \texttt{exists}\,(Q_0)$

$$
\begin{aligned}
&\textbf{Eval}\ [\![\texttt{exists}\,(Q_0) \uparrow \tbinom{k}{j}]\!]\ \rho \\
&\quad = \textbf{Eval}\ [\![\texttt{exists}\,(Q_0 \uparrow \tbinom{k}{j})]\!]\ \rho \\
&\quad = \exists\, x\ \in\ \textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ \rho \\
&\quad = \exists\, x\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \quad \text{(by induction)} \\
&\quad = \textbf{Eval}\ [\![\texttt{exists}\,(Q_0)]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q\ \equiv\ \texttt{unique}\,(Q_0)$

$$
\begin{aligned}
&\textbf{Eval}\ [\![\texttt{unique}\,(Q_0) \uparrow \tbinom{k}{j}]\!]\ \rho \\
&\quad = \textbf{Eval}\ [\![\texttt{unique}\,(Q_0 \uparrow \tbinom{k}{j})]\!]\ \rho \\
&\quad = |\textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ \rho| = 1 \\
&\quad = |\textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)| = 1 \quad \text{(by induction)} \\
&\quad = \textbf{Eval}\ [\![\texttt{unique}\,(Q_0)]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q\ \equiv\ Q_0\ \texttt{==}\ \texttt{some}\,(Q_1)$ (and similarly for $!=, <, >, \leq, \geq$)

$$
\begin{aligned}
&\textbf{Eval}\ [\![(Q_0\ \texttt{==}\ \texttt{some}\,(Q_1)) \uparrow \tbinom{k}{j}]\!]\ \rho \\
&\quad = \textbf{Eval}\ [\![(Q_0 \uparrow \tbinom{k}{j})\ \texttt{==}\ \texttt{some}\,(Q_1 \uparrow \tbinom{k}{j})]\!]\ \rho \\
&\quad = \exists\, x\ \in\ \textbf{Eval}\ [\![Q_1 \uparrow \tbinom{k}{j}]\!]\ \rho\ (\textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ \rho\ =\ x) \\
&\quad = \begin{array}{l}\exists\, x\ \in\ \textbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \\ \quad (\textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)\ =\ x)\end{array} \quad \text{(by induction)} \\
&\quad = \textbf{Eval}\ [\![Q_0\ \texttt{==}\ \texttt{some}\,(Q_1)]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

Case: $Q\ \equiv\ Q_0\ \texttt{==}\ \texttt{all}\,(Q_1)$ (and similarly for $!=, <, >, \leq, \geq$)

$$
\begin{aligned}
&\textbf{Eval}\ [\![(Q_0\ \texttt{==}\ \texttt{all}\,(Q_1)) \uparrow \tbinom{k}{j}]\!]\ \rho \\
&\quad = \textbf{Eval}\ [\![(Q_0 \uparrow \tbinom{k}{j})\ \texttt{==}\ \texttt{all}\,(Q_1 \uparrow \tbinom{k}{j})]\!]\ \rho \\
&\quad = \forall\, x\ \in\ \textbf{Eval}\ [\![Q_1 \uparrow \tbinom{k}{j}]\!]\ \rho\ (\textbf{Eval}\ [\![Q_0 \uparrow \tbinom{k}{j}]\!]\ \rho\ =\ x) \\
&\quad = \begin{array}{l}\forall\, x\ \in\ \textbf{Eval}\ [\![Q_1]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0) \\ \quad (\textbf{Eval}\ [\![Q_0]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)\ =\ x)\end{array} \quad \text{(by induction)} \\
&\quad = \textbf{Eval}\ [\![Q_0\ \texttt{==}\ \texttt{all}\,(Q_1)]\!]\ (\epsilon,\ x_n, \ldots,\ x_{j+1},\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_{j-1}, \ldots,\ x_0)
\end{aligned}
$$

□

3.2.2 OQL$_{deB}$ Identities

We can use the denotational semantics of OQL$_{deB}$ to prove identities over OQL$_{deB}$ queries. Specifically, two OQL$_{deB}$ expressions Q and Q' are equivalent (and hence substitutable for one another) iff

- *Environment Conformity:* both expressions are well-formed with respect to the same set of environments, and
- *Semantic Equivalence:* for all such environments, ρ,

$$\textbf{Eval} \ [\![Q]\!] \ \rho \quad = \quad \textbf{Eval} \ [\![Q']\!] \ \rho.$$

We prove some basic OQL$_{deB}$ identities below. For the purpose of proving any identity of OQL$_{deB}$ expressions, $Q \equiv Q'$, we refer to Q as the *lhs* (left-hand side) OQL$_{deB}$ expression and to Q' as the *rhs* (right-hand side) OQL$_{deB}$ expression.

Lemma 3.3 *For any OQL$_{deB}$ expressions Q_s and Q_w, and OQL$_{deB}$ set expressions $Q_0, \ldots, Q_i, Q_{i+1}, \ldots, Q_m$ $(0 \leq i < m)$,*

$$\texttt{flatten} \left(\begin{array}{ll} \texttt{select distinct} & \left(\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_{i+1}, \ldots, Q_m \\ \texttt{where } Q_w \end{array} \right) \\ \\ \texttt{from } Q_0, \ldots, Q_i & \end{array} \right) \equiv \begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0, \ldots, Q_m \\ \texttt{where } Q_w \end{array}$$

Proof:

Environment Conformity: Let ρ be any environment such that the lhs OQL$_{deB}$ expression is well-formed wrt ρ. Then:

- For $0 \leq j \leq m$, Q_j is well-formed wrt $(\rho, y_0, \ldots, y_{j-1})$ such that each of the values y_h $(0 \leq h \leq j-1)$ is an element of $\textbf{Eval} \ [\![Q_h]\!] \ (\rho, y_0, \ldots, y_{h-1})$.
- Q_s and Q_w are well-formed wrt $(\rho, y_0, \ldots, y_m)$.

But this makes the rhs OQL$_{deB}$ expression well-formed wrt to ρ also. Symmetric reasoning shows that the rhs OQL$_{deB}$ expression being well-formed wrt ρ implies the lhs OQL$_{deB}$ expression to be well-formed wrt ρ also.

Semantic Equivalence: Let ρ be any environment such that the lhs OQL$_{deB}$ expression (and hence the rhs OQL$_{deB}$ expression) is well-formed wrt ρ. Then we have:

$$\textbf{Eval} \left[\!\!\left[\texttt{flatten} \left(\begin{array}{ll} \texttt{select distinct} & \left(\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_{i+1}, \ldots, Q_m \\ \texttt{where } Q_w \end{array} \right) \\ \\ \texttt{from } Q_0, \ldots, Q_i & \end{array} \right) \right]\!\!\right] \rho$$

$$= \left\{ y \mid \exists x \in \left(\textbf{Eval} \left[\!\!\left[\begin{array}{ll} \texttt{select distinct} & \left(\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_{i+1}, \ldots, Q_m \\ \texttt{where } Q_w \end{array} \right) \\ \\ \texttt{from } Q_0, \ldots, Q_i & \end{array} \right]\!\!\right] \rho \right) \ (y \in x) \right\}$$

$$= \left\{ y \mid \exists x \in \left\{ \begin{array}{l} \textbf{Eval} \left[\!\!\left[\left(\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_{i+1}, \ldots, Q_m \\ \texttt{where } Q_w \end{array} \right) \right]\!\!\right] (\rho, x_0, \ldots, x_i) \mid \\ \\ x_0 \in \textbf{Eval} \ [\![Q_0]\!] \ \rho, \ldots, \\ x_i \in \textbf{Eval} \ [\![Q_i]\!] \ (\rho, x_0, \ldots, x_{i-1}) \end{array} \right\} (y \in x) \right\}$$

$$= \left\{ y \mid \exists x \in \left\{ \begin{array}{l} \left\{ \begin{array}{l} \textbf{Eval}\ [\![Q_s]\!]\ (\rho,\ x_0, \ldots,\ x_i,\ x_{i+1}, \ldots,\ x_m) \mid \\ \ x_{i+1}\ \in\ \textbf{Eval}\ [\![Q_{i+1}]\!]\ (\rho,\ x_0, \ldots,\ x_i), \ldots, \\ \ x_m\ \in\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ x_0, \ldots,\ x_i,\ x_{i+1}, \ldots,\ x_{m-1}) \\ \ \textbf{Eval}\ [\![Q_w]\!]\ (\rho,\ x_0, \ldots,\ x_i,\ x_{i+1}, \ldots,\ x_m)\ =\ \mathit{true} \end{array} \right\} \mid \\ \\ x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ \rho, \ldots, \\ x_i\ \in\ \textbf{Eval}\ [\![Q_i]\!]\ (\rho,\ x_0, \ldots,\ x_{i-1}) \end{array} \right\} (y \in x) \right\}$$

$$= \left\{ \begin{array}{l} \textbf{Eval}\ [\![Q_s]\!]\ (\rho,\ x_0, \ldots,\ x_m) \mid \\ \ x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ \rho, \ldots, \\ \ x_i\ \in\ \textbf{Eval}\ [\![Q_i]\!]\ (\rho,\ x_0, \ldots,\ x_{i-1}), \\ \ x_{i+1}\ \in\ \textbf{Eval}\ [\![Q_{i+1}]\!]\ (\rho,\ x_0, \ldots,\ x_i), \ldots, \\ \ x_m\ \in\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ x_0, \ldots,\ x_{m-1}) \\ \ \textbf{Eval}\ [\![Q_w]\!]\ (\rho,\ x_0, \ldots,\ x_m)\ =\ \mathit{true} \end{array} \right\}$$

$$= \textbf{Eval} \left[\!\!\left[\begin{array}{l} \texttt{select distinct}\ Q_s \\ \texttt{from}\ Q_0, \ldots,\ Q_i,\ Q_{i+1}, \ldots,\ Q_m \\ \texttt{where}\ Q_w \end{array} \right]\!\!\right] \rho$$

□

Lemma 3.4 *For any OQL_{deB} set expressions Q,*

$$\texttt{flatten}\ (\texttt{set}\ (Q))\ \equiv\ Q.$$

Proof:

Environment Conformity: Let ρ be any environment such that the lhs OQL_{deB} expression is well-formed wrt ρ. Then `set` (Q) is well-formed wrt ρ as is Q. Symmetric reasoning shows that the rhs OQL_{deB} expression being well-formed wrt ρ implies the lhs OQL_{deB} expression to be well-formed wrt ρ also.

Semantic Equivalence: Let ρ be any environment such that Q is well-formed wrt ρ. Then we have:

$$\begin{array}{rcl} \textbf{Eval}\ [\![\texttt{flatten}\ (\texttt{set}\ (Q))]\!]\ \rho & = & \{y \mid \exists\ x\ \in\ \textbf{Eval}\ [\![\texttt{set}\ (Q)]\!]\ \rho\ (y\ \in\ x)\} \\ & = & \{y \mid \exists\ x\ \in\ \{\textbf{Eval}\ [\![Q]\!]\ \rho\}\ (y\ \in\ x)\} \\ & = & \{y \mid y\ \in\ \textbf{Eval}\ [\![Q]\!]\ \rho\} \\ & = & \textbf{Eval}\ [\![Q]\!]\ \rho\quad \square \end{array}$$

Lemma 3.5 *For any OQL_{deB} expressions Q and Q':*

$$\begin{array}{l} \texttt{select distinct}\ Q \\ \texttt{from set}\ (Q') \end{array}\ \equiv\ \texttt{set}\ (Q \downarrow (^{Q'}_{0}))$$

Proof:

Environment Conformity: Let ρ be any environment such that the lhs OQL_{deB} expression is well-formed wrt ρ. Then

- `set` (Q') and hence Q' are well-formed wrt ρ,
- Q is well-formed wrt $(\rho,\ x)$ where $x \in$ **Eval** $[\![\texttt{set}\ (Q')]\!]\ \rho$. That is, $x \in \{\textbf{Eval}\ [\![Q']\!]\ \rho\}$ and Q is well-formed wrt $(\rho, \textbf{Eval}\ [\![Q']\!]\ \rho)$. Therefore, $Q \downarrow \binom{Q'}{0}$ is well-formed wrt ρ (by Lemma 3.2).

But that makes the rhs OQL_{deB} expression well-formed wrt ρ also. Symmetric reasoning shows that if the rhs OQL_{deB} expression is well-formed wrt some environment ρ, then so too is the lhs OQL_{deB} expression.

Semantic Equivalence: Let ρ be an environment such that the lhs OQL_{deB} expression is well-formed wrt ρ. Then we have,

$$\begin{aligned}
\textbf{Eval}\ \left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ Q\\ \texttt{from set}\ (Q')\end{array}\right]\!\!\right]\ \rho &= \{\textbf{Eval}\ [\![Q]\!]\ (\rho,\ x) \mid x \in \textbf{Eval}\ [\![\texttt{set}\ (Q')]\!]\ \rho\}\\
&= \{\textbf{Eval}\ [\![Q]\!]\ (\rho,\ x) \mid x \in \{\textbf{Eval}\ [\![Q']\!]\ \rho\}\}\\
&= \{\textbf{Eval}\ [\![Q]\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho)\}\\
&= \{\textbf{Eval}\ [\![Q \downarrow \tbinom{Q'}{0}]\!]\ \rho\} \qquad \text{By Lemma 3.2}\\
&= \textbf{Eval}\ [\![\texttt{set}\ (Q \downarrow \tbinom{Q'}{0})]\!]\ \rho \quad \square
\end{aligned}$$

Lemma 3.6 *For any OQL_{deB} expressions Q, Q' and Q'', and OQL_{deB} set expressions $Q_0,\ Q_1, \ldots,\ Q_m$ $(m \geq 0)$:*

$$\begin{array}{l}\texttt{select distinct}\ Q\\ \texttt{from set}\ (Q'),\ Q_0,\ Q_1, \ldots,\ Q_m\\ \texttt{where}\ Q''\end{array} \equiv \begin{array}{l}\texttt{select distinct}\ Q \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{m+1}\\ \texttt{from}\ Q_0 \downarrow \binom{Q'}{0},\ Q_1 \downarrow \binom{Q' \uparrow \binom{1}{0}}{1}, \ldots,\ Q_m \downarrow \binom{Q' \uparrow \binom{m}{0}}{m}\\ \texttt{where}\ Q'' \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{m+1}\end{array}$$

Proof:

Environment Conformity: Let ρ be any environment such that the lhs OQL_{deB} expression is well-formed wrt ρ. Then

- `set` (Q') and hence Q' are well-formed wrt ρ,
- Q_i $(0 \leq i \leq m)$ is well-formed wrt $(\rho,\ x,\ x_0, \ldots, x_{i-1})$ where $x \in \textbf{Eval}\ [\![\texttt{set}\ (Q')]\!]\ \rho$ and each of $x_0, \ldots,\ x_{i-1}$ is a member of

  $$\textbf{Eval}\ [\![Q_0]\!]\ (\rho,\ x), \ldots,\ \textbf{Eval}\ [\![Q_{i-1}]\!]\ (\rho,\ x,\ x_0, \ldots,\ x_{i-2})$$

  respectively. That is, $x = \textbf{Eval}\ [\![Q']\!]\ \rho$ and each of $x_0, \ldots,\ x_{i-1}$ is a member of

  $$\textbf{Eval}\ [\![Q_0]\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho), \ldots,\ \textbf{Eval}\ [\![Q_{i-1}]\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_{i-2})$$

  respectively. Therefore, each $Q' \uparrow \binom{i}{0}$ is well-formed wrt $(\rho,\ x_0, \ldots,\ x_{i-1})$ by Lemma 3.1 and each $Q_i \downarrow \binom{Q' \uparrow \binom{i}{0}}{i}$ is well-formed wrt $(\rho,\ x_0, \ldots,\ x_{i-1})$ respectively by Lemma 3.2.
- Q and Q'' are well-formed wrt $(\rho,\ x,\ x_0, \ldots, x_m)$ where $x \in \textbf{Eval}\ [\![\texttt{set}\ (Q')]\!]\ \rho$ and each of $x_0, \ldots,\ x_m$ is a member of

  $$\textbf{Eval}\ [\![Q_0]\!]\ (\rho,\ x), \ldots,\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ x,\ x_0, \ldots,\ x_{m-1})$$

respectively. That is, $x = \textbf{Eval}\ [\![Q']\!]\ \rho$ and each of $x_0, \ldots,\ x_m$ is a member of

$$\textbf{Eval}\ [\![Q_0]\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho), \ldots,\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_{m-1})$$

respectively. Therefore, $Q' \uparrow \binom{m+1}{0}$ is well-formed wrt $(\rho,\ x_0, \ldots,\ x_m)$ by Lemma 3.1 and $Q \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{m+1}$ and $Q'' \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{m+1}$ are well-formed wrt $(\rho,\ x_0, \ldots,\ x_m)$ by Lemma 3.2.

But that makes the rhs OQL_{deB} expression well-formed wrt ρ also. Symmetric reasoning shows that if the rhs OQL_{deB} expression is well-formed wrt some environment ρ, then so too is the lhs OQL_{deB} expression.

Semantic Equivalence: Let ρ be an environment such that the lhs OQL_{deB} expression is well-formed wrt ρ. Then we have,

$$\begin{aligned}
&\textbf{Eval}\ \left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ Q \\ \texttt{from set}\ (Q'),\ Q_0,\ Q_1, \ldots,\ Q_m \\ \texttt{where}\ Q''\end{array}\right]\!\!\right]\ \rho \\
= &\left\{\begin{array}{l}\textbf{Eval}\ [\![Q]\!]\ (\rho,\ x,\ x_0, \ldots,\ x_m)\ | \\ \quad x\ \in\ \textbf{Eval}\ [\![\texttt{set}\ (Q')]\!]\ \rho, \\ \quad x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\rho,\ x), \ldots, \\ \quad x_m\ \in\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ x,\ x_0, \ldots,\ x_{m-1}), \\ \textbf{Eval}\ [\![Q'']\!]\ (\rho,\ x,\ x_0, \ldots,\ x_m)\ =\ \mathit{true}\end{array}\right\} \\
= &\left\{\begin{array}{l}\textbf{Eval}\ [\![Q]\!]\ (\rho,\ x,\ x_0, \ldots,\ x_m)\ | \\ \quad x\ \in\ \{\textbf{Eval}\ [\![Q']\!]\ \rho\}, \\ \quad x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\rho,\ x), \ldots, \\ \quad x_m\ \in\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ x,\ x_0, \ldots,\ x_{m-1}), \\ \textbf{Eval}\ [\![Q'']\!]\ (\rho,\ x,\ x_0, \ldots,\ x_m)\ =\ \mathit{true}\end{array}\right\} \\
= &\left\{\begin{array}{l}\textbf{Eval}\ [\![Q]\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)\ | \\ \quad x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho), \ldots, \\ \quad x_m\ \in\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_{m-1}), \\ \textbf{Eval}\ [\![Q'']\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)\ =\ \mathit{true}\end{array}\right\} \\
= &\left\{\begin{array}{l}\textbf{Eval}\ [\![Q]\!]\ (\rho,\ \textbf{Eval}\ [\![Q' \uparrow \binom{m+1}{0}]\!]\ (\rho,\ x_0, \ldots,\ x_m),\ x_0, \ldots,\ x_m)\ | \\ \quad x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\rho,\ \textbf{Eval}\ [\![Q']\!]\ \rho), \ldots, \\ \quad x_m\ \in\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ \textbf{Eval}\ [\![Q' \uparrow \binom{m}{0}]\!]\ (\rho,\ x_0, \ldots,\ x_{m-1}),\ x_0, \ldots,\ x_{m-1}), \\ \textbf{Eval}\ [\![Q'']\!]\ (\rho,\ \textbf{Eval}\ [\![Q' \uparrow \binom{m+1}{0}]\!]\ (\rho,\ x_0, \ldots,\ x_m),\ x_0, \ldots,\ x_m)\ =\ \mathit{true}\end{array}\right\}\ \text{(By Lemma 3.1)} \\
= &\left\{\begin{array}{l}\textbf{Eval}\ \left[\!\!\left[Q \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{m+1}\right]\!\!\right]\ (\rho,\ x_0, \ldots,\ x_m)\ | \\ \quad x_0\ \in\ \textbf{Eval}\ [\![Q_0 \downarrow \binom{Q'}{0}]\!]\ \rho, \ldots, \\ \quad x_m\ \in\ \textbf{Eval}\ \left[\!\!\left[Q_m \downarrow \binom{Q' \uparrow \binom{m}{0}}{m}\right]\!\!\right]\ (\rho,\ x_0, \ldots,\ x_{m-1}), \\ \textbf{Eval}\ \left[\!\!\left[Q'' \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{m+1}\right]\!\!\right]\ (\rho,\ x_0, \ldots,\ x_m)\ =\ \mathit{true}\end{array}\right\}\ \text{(By Lemma 3.2)} \\
= &\ \textbf{Eval}\ \left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ Q \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{m+1} \\ \texttt{from}\ Q_0 \downarrow \binom{Q'}{0}, \ldots,\ Q_m \downarrow \binom{Q' \uparrow \binom{m}{0}}{m} \\ \texttt{where}\ Q'' \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{m+1}\end{array}\right]\!\!\right]\ \rho\ \ \square
\end{aligned}$$

Lemma 3.7 *For any OQL_{deB} expressions Q and Q' and OQL_{deB} set expressions $Q_0, \ldots,\ Q_{m+1}$ ($m \geq 0$):*

$$\begin{array}{l}\texttt{select distinct}\ Q \\ \texttt{from}\ \left(\begin{array}{l}\texttt{select distinct}\ Q' \\ \texttt{from}\ Q_0, \ldots,\ Q_m\end{array}\right),\ Q_{m+1}\end{array} \quad \equiv \quad \begin{array}{l}\texttt{select distinct}\ (Q \uparrow \binom{m+1}{2}) \downarrow \binom{Q' \uparrow \binom{1}{0}}{1} \\ \texttt{from}\ Q_0, \ldots,\ Q_m,\ (Q_{m+1} \uparrow \binom{m+1}{1}) \downarrow \binom{Q'}{0}\end{array}$$

Proof:

Environment Conformity: Let ρ be any environment such that the lhs OQL$_{deB}$ expression is well-formed wrt ρ. Then:

- For $0 \leq j \leq m$, Q_j is well-formed wrt $(\rho,\ y_0, \ldots,\ y_{j-1})$ such that each value y_h is an element of $\textbf{Eval}\ [\![Q_h]\!]\ (\rho,\ y_0, \ldots,\ y_{h-1})$.

- Q_{m+1} is well-formed wrt $(\rho,\ x)$ such that x is an element of

$$\textbf{Eval}\ \left[\!\!\left[\begin{array}{l} \texttt{select distinct } Q' \\ \texttt{from } Q_0, \ldots,\ Q_m \end{array} \right]\!\!\right] \rho$$

  That is, x is of the form $\textbf{Eval}\ [\![Q']\!]\ (\rho,\ y_0, \ldots,\ y_m)$ for $y_0, \ldots, y_m \in$

$$\textbf{Eval}\ [\![Q_0]\!]\ \rho, \ldots,\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ y_0, \ldots,\ y_{m-1})$$

  respectively. Therefore, $Q_{m+1} \uparrow \binom{m+1}{1}$ is well-formed wrt

$$(\rho,\ y_0, \ldots,\ y_m,\ \textbf{Eval}\ [\![Q']\!]\ (\rho,\ y_0, \ldots,\ y_m))$$

  (by Lemma 3.1) and $(Q_{m+1} \uparrow \binom{m+1}{1}) \downarrow \binom{Q'}{0}$ is well-formed wrt

$$(\rho,\ y_0, \ldots,\ y_m)$$

  by Lemma 3.2.

- Q' is well-formed wrt $(\rho,\ y_0, \ldots,\ y_m)$. Therefore, $Q' \uparrow \binom{1}{0}$ is well-formed wrt $(\rho,\ y_0, \ldots,\ y_m,\ y_{m+1})$ by Lemma 3.1.

- Q is well-formed wrt $(\rho,\ x,\ y_{m+1})$ such that x is an element of

$$\textbf{Eval}\ \left[\!\!\left[\begin{array}{l} \texttt{select distinct } Q' \\ \texttt{from } Q_0, \ldots,\ Q_m \end{array} \right]\!\!\right] \rho,$$

  That is, x is of the form $\textbf{Eval}\ [\![Q']\!]\ (\rho,\ y_0, \ldots,\ y_m)$ for $y_0, \ldots, y_m \in$

$$\textbf{Eval}\ [\![Q_0]\!]\ \rho, \ldots,\ \textbf{Eval}\ [\![Q_m]\!]\ (\rho,\ y_0, \ldots,\ y_{m-1})$$

  respectively, and y_{m+1} is an element of $\textbf{Eval}\ [\![Q_{m+1}]\!]\ (\rho,\ x)$. Therefore, Q is well-formed wrt

$$(\rho,\ \textbf{Eval}\ [\![Q']\!]\ (\rho,\ y_0, \ldots,\ y_m),\ y_{m+1}),$$

  $Q \uparrow \binom{m+1}{2}$ is well-formed wrt

$$(\rho,\ y_0, \ldots,\ y_m,\ \textbf{Eval}\ [\![Q' \uparrow \tbinom{1}{0}]\!]\ (\rho,\ y_0, \ldots,\ y_m,\ y_{m+1}),\ y_{m+1})$$

  by Lemma 3.1, and $(Q \uparrow \binom{m+1}{2}) \downarrow \binom{Q' \uparrow \binom{1}{0}}{1}$ is well-formed wrt

$$(\rho,\ y_0, \ldots,\ y_m,\ y_{m+1})$$

  by Lemma 3.2.

Thus, the rhs OQL$_{deB}$ expression is well-formed wrt ρ. Symmetric reasoning establishes that for any environment ρ such that the rhs OQL$_{deB}$ expression is well-formed wrt ρ, the lhs OQL$_{deB}$ expression is also well-formed wrt ρ.

Semantic Equivalence: Let ρ be any environment that makes the lhs OQL$_{deB}$ expression (and hence the rhs OQL$_{deB}$ expression) well-formed. Then we have:

$$\mathbf{Eval}\left[\!\!\left[\begin{array}{l}\texttt{select distinct } Q\\ \texttt{from}\left(\begin{array}{l}\texttt{select distinct } Q'\\ \texttt{from } Q_0,\ldots,\ Q_m\end{array}\right),\ Q_{m+1}\end{array}\right]\!\!\right]\rho$$

$$= \left\{\ \mathbf{Eval}\ [\![Q]\!]\ (\rho,\ x,\ y_{m+1})\ \middle|\ \begin{array}{l} x\ \in\ \mathbf{Eval}\left[\!\!\left[\begin{array}{l}\texttt{select distinct } Q'\\ \texttt{from } Q_0,\ldots,\ Q_m\end{array}\right]\!\!\right]\rho,\\ y_{m+1}\in \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\rho,\ x)\end{array}\right\}$$

$$= \left\{\ \mathbf{Eval}\ [\![Q]\!]\ (\rho,\ x,\ y_{m+1})\ \middle|\ \begin{array}{l} x\ \in\ \left\{\begin{array}{l}\mathbf{Eval}\ [\![Q']\!]\ (\rho,\ y_0,\ldots,\ y_m)\ |\\ \quad y_0\ \in\ \mathbf{Eval}\ [\![Q_0]\!]\ \rho,\ldots,\\ \quad y_m\ \in\ \mathbf{Eval}\ [\![Q_m]\!]\ (\rho,\ y_0,\ldots,\ y_{m-1})\end{array}\right\},\\ y_{m+1}\ \in\ \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\rho,\ x)\end{array}\right\}$$

$$= \left\{\begin{array}{l}\mathbf{Eval}\ [\![Q]\!]\ (\rho,\ \mathbf{Eval}\ [\![Q']\!]\ (\rho,\ y_0,\ldots,\ y_m),\ y_{m+1})\ |\\ \quad y_0\ \in\ \mathbf{Eval}\ [\![Q_0]\!]\ \rho,\ldots,\\ \quad y_m\ \in\ \mathbf{Eval}\ [\![Q_m]\!]\ (\rho,\ y_0,\ldots,\ y_{m-1}),\\ \quad y_{m+1}\ \in\ \mathbf{Eval}\ [\![Q_{m+1}]\!]\ (\rho,\ \mathbf{Eval}\ [\![Q']\!]\ (\rho,\ y_0,\ldots,\ y_m))\end{array}\right\}$$

$$= \left\{\begin{array}{l}\mathbf{Eval}\ [\![Q\uparrow\binom{m+1}{2}]\!]\ (\rho,\ y_0,\ldots,\ y_m,\ \mathbf{Eval}\ [\![Q'\uparrow\binom{1}{0}]\!]\ (\rho,\ y_0,\ldots,\ y_m,\ y_{m+1}),\ y_{m+1})\ |\\ \quad y_0\ \in\ \mathbf{Eval}\ [\![Q_0]\!]\ \rho,\ldots,\\ \quad y_m\ \in\ \mathbf{Eval}\ [\![Q_m]\!]\ (\rho,\ y_0,\ldots,\ y_{m-1}),\\ \quad y_{m+1}\ \in\ \mathbf{Eval}\ [\![Q_{m+1}\uparrow\binom{m+1}{1}]\!]\ (\rho,\ y_0,\ldots,\ y_m,\ \mathbf{Eval}\ [\![Q']\!]\ (\rho,\ y_0,\ldots,\ y_m))\end{array}\right\}\ \text{(by Lemma 3.1)}$$

$$= \left\{\begin{array}{l}\mathbf{Eval}\ \left[\!\!\left[(Q\uparrow\binom{m+1}{2})\downarrow\binom{Q'\uparrow\binom{1}{0}}{1}\right]\!\!\right]\ (\rho,\ y_0,\ldots,\ y_m,\ y_{m+1})\ |\\ \quad y_0\ \in\ \mathbf{Eval}\ [\![Q_0]\!]\ \rho,\ldots,\\ \quad y_m\ \in\ \mathbf{Eval}\ [\![Q_m]\!]\ (\rho,\ y_0,\ldots,\ y_{m-1}),\\ \quad y_{m+1}\ \in\ \mathbf{Eval}\ [\![(Q_{m+1}\uparrow\binom{m+1}{1})\downarrow\binom{Q'}{0}]\!]\ (\rho,\ y_0,\ldots,\ y_m)\end{array}\right\}\ \text{(by Lemma 3.2)}$$

$$= \mathbf{Eval}\left[\!\!\left[\begin{array}{l}\texttt{select distinct } (Q\uparrow\binom{m+1}{2})\downarrow\binom{Q'\uparrow\binom{1}{0}}{1}\\ \texttt{from } Q_0,\ldots,\ Q_m,\ (Q_{m+1}\uparrow\binom{m+1}{1})\downarrow\binom{Q'}{0}\end{array}\right]\!\!\right]\rho\ \ \square$$

Lemma 3.8 *For any OQL expressions Q, Q' and Q'' and set expressions $Q_0,\ldots,\ Q_m$ $(m\geq 0)$:*

$$\begin{array}{l}\texttt{select distinct } Q\\ \texttt{from}\left(\begin{array}{l}\texttt{select distinct } Q'\\ \texttt{from } Q_0,\ldots,\ Q_m\end{array}\right)\\ \texttt{where } Q''\end{array}\ \equiv\ \begin{array}{l}\texttt{select distinct } (Q\uparrow\binom{m+1}{1})\downarrow\binom{Q'}{0}\\ \texttt{from } Q_0,\ldots,\ Q_m\\ \texttt{where } (Q''\uparrow\binom{m+1}{1})\downarrow\binom{Q'}{0}\end{array}$$

Proof:

Environment Conformity: Let ρ be any environment such that the lhs OQL$_{deB}$ expression is well-formed wrt ρ. Then:

- For $0\leq j\leq m$, Q_j is well-formed wrt $(\rho,\ y_0,\ldots,\ y_{j-1})$ such that each value y_h is an element of $\mathbf{Eval}\ [\![Q_h]\!]\ (\rho,\ y_0,\ldots,\ y_{h-1})$.
- Q' is well-formed with respect to $(\rho,\ y_0,\ldots,\ y_m)$.
- Q and Q'' are well-formed wrt $(\rho,\ x)$, where

$$x\in\mathbf{Eval}\left[\!\!\left[\begin{array}{l}\texttt{select distinct } Q'\\ \texttt{from } Q_0,\ldots,\ Q_m\end{array}\right]\!\!\right]\rho.$$

That is, Q and Q'' are well-formed wrt $(\rho,\ \textbf{Eval}\ \llbracket Q' \rrbracket\ (\rho,\ y_0, \ldots,\ y_m))$ where each of $y_0, \ldots,\ y_m$ is in

$$\textbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho, \ldots,\ \textbf{Eval}\ \llbracket Q_m \rrbracket\ (\rho,\ y_0, \ldots,\ y_{m-1})$$

respectively. Therefore, $Q \uparrow \tbinom{m+1}{1}$ and $Q \uparrow \tbinom{m+1}{1}$ are well-formed wrt

$$(\rho,\ y_0, \ldots,\ y_m,\ \textbf{Eval}\ \llbracket Q' \rrbracket\ (\rho,\ y_0, \ldots,\ y_m))$$

by Lemma 3.1 and $(Q \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q'}{0}$ and $(Q'' \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q'}{0}$ are well-formed wrt $(\rho,\ y_0, \ldots,\ y_m)$ by Lemma 3.2.

Therefore, the rhs OQL_{deB} expression is also well-formed wrt ρ. Symmetric reasoning establishes that for any environment ρ such that the rhs OQL_{deB} expression is well-formed wrt ρ, the lhs OQL_{deB} expression is also well-formed wrt ρ.

Semantic Equivalence: Let ρ be an environment that makes the left-hand expression well-formed. Then we have:

$$
\begin{aligned}
&\textbf{Eval} \left\llbracket \begin{array}{l} \texttt{select distinct } Q \\ \texttt{from} \left(\begin{array}{l} \texttt{select distinct } Q' \\ \texttt{from } Q_0, \ldots,\ Q_m \end{array} \right) \\ \texttt{where } Q'' \end{array} \right\rrbracket \rho \\
&= \left\{ \textbf{Eval}\ \llbracket Q \rrbracket\ (\rho,\ x) \mid x \in \textbf{Eval} \left\llbracket \begin{array}{l} \texttt{select distinct } Q' \\ \texttt{from } Q_0, \ldots,\ Q_m \end{array} \right\rrbracket \rho,\ \textbf{Eval}\ \llbracket Q'' \rrbracket\ (\rho,\ x) = \mathit{true} \right\} \\
&= \left\{ \textbf{Eval}\ \llbracket Q \rrbracket\ (\rho,\ x) \mid x \in \left\{ \begin{array}{l} \textbf{Eval}\ \llbracket Q' \rrbracket\ (\rho,\ y_0, \ldots,\ y_m) \mid \\ \quad y_0 \in \textbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho, \ldots, \\ \quad y_m \in \textbf{Eval}\ \llbracket Q_m \rrbracket\ (\rho,\ y_0, \ldots,\ y_{m-1}) \end{array} \right\},\ \textbf{Eval}\ \llbracket Q'' \rrbracket\ (\rho,\ x) = \mathit{true} \right\} \\
&= \left\{ \begin{array}{l} \textbf{Eval}\ \llbracket Q \rrbracket\ (\rho,\ \textbf{Eval}\ \llbracket Q' \rrbracket\ (\rho,\ y_0, \ldots,\ y_m)) \mid \\ \quad y_0 \in \textbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho, \ldots, \\ \quad y_m \in \textbf{Eval}\ \llbracket Q_m \rrbracket\ (\rho,\ y_0, \ldots,\ y_{m-1}), \\ \quad \textbf{Eval}\ \llbracket Q'' \rrbracket\ (\rho,\ \textbf{Eval}\ \llbracket Q' \rrbracket\ (\rho,\ y_0, \ldots,\ y_m)) = \mathit{true} \end{array} \right\} \\
&= \left\{ \begin{array}{l} \textbf{Eval}\ \llbracket Q \uparrow \tbinom{m+1}{1} \rrbracket\ (\rho,\ y_0, \ldots,\ y_m,\ \textbf{Eval}\ \llbracket Q' \rrbracket\ (\rho,\ y_0, \ldots,\ y_m)) \mid \\ \quad y_0 \in \textbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho, \ldots, \\ \quad y_m \in \textbf{Eval}\ \llbracket Q_m \rrbracket\ (\rho,\ y_0, \ldots,\ y_{m-1}), \\ \quad \textbf{Eval}\ \llbracket Q'' \uparrow \tbinom{m+1}{1} \rrbracket\ (\rho,\ y_0, \ldots,\ y_m,\ \textbf{Eval}\ \llbracket Q' \rrbracket\ (\rho,\ y_0, \ldots,\ y_m)) = \mathit{true} \end{array} \right\} \\
&= \left\{ \begin{array}{l} \textbf{Eval}\ \left\llbracket (Q \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q'}{0} \right\rrbracket\ (\rho,\ y_0, \ldots,\ y_m) \mid \\ \quad y_0 \in \textbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho, \ldots, \\ \quad y_m \in \textbf{Eval}\ \llbracket Q_m \rrbracket\ (\rho,\ y_0, \ldots,\ y_{m-1}), \\ \quad \textbf{Eval}\ \left\llbracket (Q'' \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q'}{0} \right\rrbracket\ (\rho,\ y_0, \ldots,\ y_m) = \mathit{true} \end{array} \right\} \\
&= \textbf{Eval} \left\llbracket \begin{array}{l} \texttt{select distinct } (Q \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q'}{0} \\ \texttt{from } Q_0, \ldots,\ Q_m \\ \texttt{where } (Q'' \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q'}{0} \end{array} \right\rrbracket \rho \quad \Box
\end{aligned}
$$

4 KOLA

4.1 KOLA

KOLA is a combinator-style (variable-free) query algebra and is the target of our translation. The exclusion of variables from function and predicate definitions simplifies their composition and decomposition and makes code supplements to rules unnecessary [CZ96].

Every KOLA function and predicate can be thought of as an OQL_{deB} macro. The invocation of the function or predicate on an object (via invocation operators __ ! __ (for functions) and __ ? __ (for predicates)) has a semantics expressible in OQL_{deB}. We provide such an operational semantics for KOLA in Table 2.

KOLA functions and predicates can be subdivided into *primitives* and *formers*. Primitive functions and predicates have a predefined semantics. For example, the first two sections of Table 2 define the semantics of KOLA primitive functions and predicates respectively. KOLA function primitives include the identity function (**id**), projection functions on pairs (π_1 and π_2), arithmetic operations on integers (**add**, **sub**, **mul**, **div**, **mod** and **abs**), and operators on sets (**ins** (insert), **set** (singleton constructor), **choose** (element extraction) and **flat** (nested set flattening)). The KOLA predicate primitives listed are comparison operators for equality (**eq**) and inequality (**neq**) of pairs of objects of any like type, and "less than" (**lt**), "greater than" (**gt**), "less than or equal" (**leq**) and "greater than or equal" (**geq**) for pairs of integers. KOLA also permits primitives to be schema-based (i.e., object *attributes*) such as an `is_married` predicate or an `age` function on a user-defined type, `Person`.)

All KOLA functions and predicates are unary. Binary functions and predicates are simulated by demanding the single argument to a function or predicate be a pair. Note that pair arguments to most KOLA operators differ from those resulting from normalizing tuples. All normalized tuples are pairs,

$$\ll P,\ Q \gg$$

whose value for P is another pair or the "empty pair", $\diamond$. Note that with the exception of projection functions (π_1 and π_2) and equality (**eq**) and inequality (**neq**) predicates, such pairs would be type-incorrect arguments to any other KOLA primitives.

KOLA formers are second-order functionals that build complex functions and predicates from simpler ones. The third through fifth sections of Table 2 define the subset of KOLA's function and predicate formers that are necessary for the translation of OQL_{deB}. (The description of the full algebra can be found in [Che95a] and its formal specification (written in Larch [GHG+92]) is available at [Che95b].) General function formers (third section) include function composition (__ $\circ$ __), function pairing ($\langle$__, __$\rangle$), constant functions ($\texttt{K}_f$ (__)), curried functions ($\texttt{C}_f$ (__, __)), pairwise function application (__ $\times$ __), and conditional functions (**con** (__, __, __)). General predicate formers (fourth section) include predicate/function combination (__ $\oplus$ __), predicate conjunction (__ & __), predicate disjunction (__ | __), predicate negation ($\sim$ (__)), constant predicates ($\texttt{K}_p$ (__)) and curried predicates ($\texttt{C}_p$ (__, __)). These formers are defined in Table 2 under the assumption that f and g are (any) functions, p and q are predicates, A and B are sets, i and j are integers, and x and y are objects or values of any type.

KOLA also includes specialized predicate and function formers that build functions and predicates over sets (*query formers*). These are listed in the fifth section of Table 2. **iterate** is similar to OQL_{deB}'s `select-distinct-from-where` operator. Given a function f and predicate p, the query function **iterate** $(p,\ f)$ applies f to objects in a set that satisfy p.

iter is similar to **iterate**, but is invoked on pairs $\ll x,\ A \gg$ (where A is a set) rather than on sets alone. **iter** permits expression of loop computations involving a constant, x, whose value is computed outside of the loop but used within. As we showed in [CZ96], this is useful for expressing nested queries as `x` can denote an OQL_{deB} expression that evaluates to an environment and that is evaluated in the body of an outer loop but used within the body of an inner loop. As we showed in [CZLK96], **iter** is also useful for expressing *code motion* transformations that move constant computations outside of loops.

KOLA includes generalized set operator formers, **int** (eq), **uni** (eq) and **dif** (eq). When instantiated with an equivalence relation, eq[4], these formers generate generalized versions of set intersection, union and

[4] An equivalence relation is a (binary) predicate that is *reflexive*, *symmetric* and *transitive*. A prototypical equivalence

difference respectively that are relative to *eq*. When instantiated with the equality primitive (**eq**), these formers generate standard intersection, union and difference functions. A full description of, and motivation for these formers is given in [CZN95].

bound is a former of partial aggregate functions and therefore is purposely underspecified. **bound** (p) generates functions that compute least upper or greatest lower bounds (wrt the *partial order predicate*, p[5]) of non-empty sets. Thus, this former captures a class of aggregates that are well-defined only for non-empty sets such as OQL's `min` and `max`.

OQL_{deB}'s other aggregates `count`, `sum` and `avg` cannot be expressed in KOLA with **bound**. Instead, they are defined using the KOLA former, **fold**, which is defined for object x, set A, function f and commutative function on pairs, $\otimes$, as:

$$\mathbf{fold}\ (x,\ f,\ op)\ !\ A \ = \ x \otimes (f\ !\ x_0) \otimes \ \ldots\ \otimes (f\ !\ x_n),$$

such that $A = \{x_0, \ldots, x_n\}$ and $x_i = x_j \Rightarrow i = j$. Effectively, **fold** applies f to each member of A and combines the results with $\otimes$. x defines the identity value returned when A is empty. (Note that $\otimes$ must be commutative and associative for **fold** to be well-defined. Unlike [BTBN91], we do not demand that $\otimes$ also be idempotent, and instead define the semantics of **fold** in terms of minimally represented sets.) Table 1 shows how OQL's partial and total aggregate functions are expressed in KOLA.

fold is inexpressible in OQL_{deB}. Instead, we introduce the KOLA "macros": $\overline{\texttt{count}}$, $\overline{\texttt{sum}}$ and $\overline{\texttt{avg}}$ defined as:

$$\begin{aligned}
\overline{\texttt{count}} &\equiv \mathbf{fold}\ (\texttt{0},\ \texttt{K}_f\ (\texttt{1}),\ \mathbf{add}) \\
\overline{\texttt{sum}} &\equiv \mathbf{fold}\ (\texttt{0},\ \mathbf{id},\ \mathbf{add}) \\
\overline{\texttt{avg}} &\equiv \mathbf{con}\ (\texttt{C}_p\ (\mathbf{eq},\ \{\}),\ 0,\ \mathbf{div} \circ \langle \overline{\texttt{sum}},\ \overline{\texttt{count}} \rangle).
\end{aligned}$$

That these macros are *correct* (i.e., that for any OQL_{deB} set expression, Q that is well-formed wrt environment ρ:

$$\mathbf{Eval}\ [\![\overline{\texttt{count}}\ !\ Q]\!]\ \rho \ = \ \mathbf{Eval}\ [\![\texttt{count}\ (Q)]\!]\ \rho$$

and for integer set Q that is well-formed wrt environment ρ:

$$\begin{aligned}
\mathbf{Eval}\ [\![\overline{\texttt{sum}}\ !\ Q]\!]\ \rho &= \mathbf{Eval}\ [\![\texttt{sum}\ (Q)]\!]\ \rho \\
\mathbf{Eval}\ [\![\overline{\texttt{avg}}\ !\ Q]\!]\ \rho &= \mathbf{Eval}\ [\![\texttt{avg}\ (Q)]\!]\ \rho)
\end{aligned}$$

is easily shown by induction on the length of **Eval** $[\![Q]\!]\ \rho$. We leave this as an exercise for the reader.

grp generalizes OQL's [Cat93] "`group by`" operator. **grp** $(p,\ f,\ g)\ !\ A$ returns a set of values, $f\ !\ \ll x,\ F_x \gg$ for each $x \in A$. For each such pair, F_x is the set resulting from applying g to every element $y \in A$ such that $p\ ?\ \ll x,\ y \gg$. To illustrate, the OQL_{deB} query below partitions a set on the basis of predicates $p_1, \ldots, p_n$.

select distinct Q_s
from Q
where Q_w
groupby $P_0, \ldots, P_n$

---

relation is

$$(\mathbf{eq} \oplus (f_1 \times f_1))\ \&\ \ldots\ \&\ (\mathbf{eq} \oplus (f_n \times f_n))$$

for any functions $f_1, \ldots, f_n$.

[5] A partial order predicate is *reflexive*, *antisymmetric* and *transitive*. Example partial order predicates are **leq** and **geq**.

AGG (OQL)	$\overline{\texttt{AGG}}$ (KOLA)
MAX	**bound** (**geq**)
MIN	**bound** (**leq**)
COUNT	**fold** (0, $\texttt{K}_f$ (1), **add**)
SUM	**fold** (0, **id**, **add**)
AVG	**con** ($\texttt{C}_p$ (**eq**, {}), $\texttt{K}_f$ (0), **div** $\circ$ $\langle\overline{\texttt{SUM}}$, $\overline{\texttt{COUNT}}\rangle$)

Table 1: OQL Aggregates (AGG) and their KOLA equivalents ($\overline{\texttt{AGG}}$)

The KOLA version of this query first creates a set of tuples of the form,

$$\ll Q_0, \ldots,\ Q_n,\ r \gg$$

such that r evaluates to a value in the set denoted by Q, and each expression, Q_i evaluates to either 1 or 0 depending on whether r satisfies predicate P_i. Then this set is grouped according to equivalence on all of the values for $Q_0, \ldots,\ Q_n$. We will return to the discussion of translating OQL$_{deB}$ queries into KOLA in Section 4.2.

KOLA's query *predicate* formers are quantifier-inspired. When instantiated with predicates and invoked on sets, **exists** and **forall** behave like existential and universal quantifiers respectively. Specifically, given some predicate p, **exists** (p) (**forall** (p)) returns *true* when invoked on a set input A if some (all) element(s) of A satisfy p.

EX and **FA** are also quantifier-inspired formers. But the predicates they generate incorporate constants into their loop predicates making them related to **EX** and **FA** as **iterate** is related to **iter**.

4.2 T: A Combinator Translation of Queries

The translation function **T** maps an OQL$_{deB}$ expression Q (that is well-formed wrt to an environment ρ) to a KOLA function, **T** $[\![Q]\!]$. If Q' is an OQL$_{deB}$ expression whose semantics is ρ (i.e., **Eval** $[\![Q']\!]\ \epsilon\ =\ \rho$), then the result of invoking **T** $[\![Q]\!]$ on Q' is an OQL expression that is well-formed wrt ϵ, and with the same semantics as Q wrt ρ.

As an example, let $Q\ =\ V_0\ +\ V_1$, which is well-formed wrt $\rho\ =\ (\epsilon,\ 3, 5)$ (and equal to 8), and let $Q'\ =\ \ll \diamond,\ 3,\ 5 \gg$ Note that the semantics of Q' is ρ. The translation of Q (**T** $[\![Q]\!]$) is **add** $\circ$ $\langle\pi_2,\ \pi_2\ \circ\ \pi_1\rangle$, which when invoked on Q' gives us,

$$\begin{aligned}
&(\textbf{add}\ \circ\ \langle\pi_2,\ \pi_2\ \circ\ \pi_1\rangle)\ !\ \ll \diamond,\ 3,\ 5 \gg\\
&=\ \textbf{add}\ !\ (\langle\pi_2,\ \pi_2\ \circ\ \pi_1\rangle\ !\ \ll \diamond,\ 3,\ 5 \gg)\\
&=\ \textbf{add}\ !\ \ll \pi_2\ !\ \ll \diamond,\ 3,\ 5 \gg,\ \pi_2\ !\ (\pi_1\ !\ \ll \diamond,\ 3,\ 5 \gg) \gg\\
&=\ \textbf{add}\ !\ \ll \ll \diamond,\ 3,\ 5 \gg\ .\ a_1,\ (\ll \diamond,\ 3,\ 5 \gg\ .\ a_0)\ .\ a_1 \gg\\
&=\ \textbf{add}\ !\ \ll 5,\ 3 \gg\\
&=\ 5\ +\ 3
\end{aligned}$$

The latter OQL$_{deB}$ expression is well-formed wrt to all environments (including ϵ) and has semantics 8.

4.2.1 Translation Described

Intuitively, translation takes one of three forms:

1. *Direct Translations:* The simplest translation is of OQL_{deB} expressions that have KOLA counterparts that are formers. For example, the translations of OQL expressions, `if` Q_0 `then` Q_1 `else` Q_2 and $Q_0 \ \wedge \ Q_1$ fully resemble the original OQL expressions.

2. *Operator + Argument Translations:* Most translations construct functions of the form $f \circ g$, where f is a KOLA primitive that corresponds to an OQL_{deB} operator in the translated expression, and g is a more complex function that prepares the arguments for the KOLA primitive. A straightforward example of this was shown earlier with the translation of an OQL_{deB} expression that added two subexpressions. The KOLA primitive **add** corresponds to the OQL addition operator (_ + _), while the pairing function $\langle \mathbf{T}\ [\![Q_0]\!],\ \mathbf{T}\ [\![Q_1]\!]\rangle$ prepares the two arguments to **add**. Other translations of this form are those of expressions involving other arithmetic operaters ($-$, $*$, $/$, `mod`, `abs`), set operators (`element`, `flatten`, `intersect`, `union`, `except`), aggregates (`count`, `avg`, `sum`, `max`, `min`), comparison predicates ($==$, $!=$, $<$, $>$, $\leq$, $\geq$, `in`) and quantifiers (`exists`, `for_all`, `unique`, `some`, `all`).

   A trickier example of this form of translation is the translation of `select−distinct−from−where` queries. The translation of

$$\mathbf{T}\left[\!\!\left[\begin{array}{l}\texttt{select distinct } Q_s\\ \texttt{from } Q_0, \ldots, Q_m\\ \texttt{where } Q_w\end{array}\right]\!\!\right]$$

   is the function, $f \circ g$, where $f = \textbf{iterate}\ (\mathbf{T}\ [\![Q_w]\!],\ \mathbf{T}\ [\![Q_s]\!])$ and $g =$

$$\begin{array}{l}\textbf{flat} \circ \textbf{iterate}\ (\texttt{K}_p\ (\texttt{T}),\ \textbf{iter}\ (\texttt{K}_p\ (\texttt{T}),\ \textbf{id}) \circ \langle \textbf{id},\ \mathbf{T}\ [\![Q_m]\!]\rangle) \circ\\ \ldots \circ\\ \textbf{flat} \circ \textbf{iterate}\ (\texttt{K}_p\ (\texttt{T}),\ \textbf{iter}\ (\texttt{K}_p\ (\texttt{T}),\ \textbf{id}) \circ \langle \textbf{id},\ \mathbf{T}\ [\![Q_0]\!]\rangle) \circ\\ \textbf{set}\end{array}$$

   When invoked on an expression Q' with semantics ρ, g has the effect of producing an OQL expression whose semantics is a set of values $(\rho,\ x_m, \ldots,\ x_0)$ where each x_i is drawn from the set Q_i $(0 \leq i \leq m)$. (Lemma 4.2 proves this formally.) When invoked on this expression, $\textbf{iterate}\ (\mathbf{T}\ [\![Q_w]\!],\ \mathbf{T}\ [\![Q_s]\!])$ has the effect of selecting and projecting on this result.

3. *Tuple Extraction and Variable Translations* The last translation worth mentioning is that of tuples and expressions which extract fields from tuples. The normalization of a tuple (as described in Section 2.1.2) constructs an OQL_{deB} expression whose evaluation results in an environment. As variables reference values in environments, it should not be surprising that the translation of an expression that extracts a field from a tuple resembles the translation of a variable which extracts a value from an environment. Both translations produce compositions of projection functions (π_1 and π_2) that when invoked on nested pairs, extract the appropriate values.

4.2.2 Translation Illustrated

To illustrate how translation proceeds, we trace the translation of the OQL query shown in Figure 1. Normalized and expressed using deBruijn notation, this query is

$$\begin{array}{l}\texttt{select distinct} \ll \diamond,\ V_0,\ \left(\begin{array}{l}\texttt{select distinct } V_0\\ \texttt{from } (V_0 \rightarrow \texttt{child})\\ \texttt{where } (V_1 \rightarrow \texttt{addr})\ ==\ (V_0 \rightarrow \texttt{addr})\end{array}\right) \gg\\ \texttt{from P}\end{array}$$

The translation of this query is traced below.

$$\mathbf{T}\left[\!\!\left[\begin{array}{l}\texttt{select distinct} \ll \diamond,\ V_0,\ \left(\begin{array}{l}\texttt{select distinct } V_0\\ \texttt{from } (V_0 \to \texttt{child})\\ \texttt{where } (V_1 \to \texttt{addr})\ \texttt{==}\ (V_0 \to \texttt{addr})\end{array}\right) \gg \\ \texttt{from P}\end{array}\right]\!\!\right]$$

$$\equiv \begin{array}{l} \mathbf{iterate}\left(\mathbf{T}\,[\![\texttt{T}]\!],\ \mathbf{T}\left[\!\!\left[\ll\diamond,\ V_0,\ \left(\begin{array}{l}\texttt{select distinct } V_0\\ \texttt{from } (V_0 \to \texttt{child})\\ \texttt{where } (V_1 \to \texttt{addr})\ \texttt{==}\ (V_0 \to \texttt{addr})\end{array}\right)\gg\right]\!\!\right]\right)\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \mathbf{T}\,[\![\texttt{P}]\!]\rangle)\circ\\ \mathbf{set}\end{array}$$

$$\equiv \begin{array}{l} \mathbf{iterate}\left(\texttt{K}_p\ (\texttt{T}),\ \left\langle\langle\texttt{K}_f\ (\diamond),\ \mathbf{T}\,[\![V_0]\!]\rangle,\ \mathbf{T}\left[\!\!\left[\begin{array}{l}\texttt{select distinct } V_0\\ \texttt{from } (V_0 \to \texttt{child})\\ \texttt{where } (V_1 \to \texttt{addr})\ \texttt{==}\ (V_0 \to \texttt{addr})\end{array}\right]\!\!\right]\right\rangle\right)\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \texttt{K}_f\ (\texttt{P})\rangle)\circ\\ \mathbf{set}\end{array}$$

$$\equiv \begin{array}{l} \mathbf{iterate}\left(\texttt{K}_p\ (\texttt{T}),\ \left\langle\langle\texttt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l}\mathbf{iterate}\,(\mathbf{T}\,[\![(V_1 \to \texttt{addr})\ \texttt{==}\ (V_0 \to \texttt{addr})]\!],\ \mathbf{T}\,[\![V_0]\!])\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \mathbf{T}\,[\![(V_0\to\texttt{child})]\!]\rangle)\circ\\ \mathbf{set}\end{array}\right\rangle\right)\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \texttt{K}_f\ (\texttt{P})\rangle)\circ\\ \mathbf{set}\end{array}$$

$$\equiv \begin{array}{l} \mathbf{iterate}\left(\texttt{K}_p\ (\texttt{T}),\ \left\langle\langle\texttt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l}\mathbf{iterate}\,(\mathbf{eq}\ \oplus\ \langle\mathbf{T}\,[\![(V_1 \to \texttt{addr})]\!],\ \mathbf{T}\,[\![(V_0 \to \texttt{addr})]\!]\rangle,\ \pi_2)\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \texttt{child}\circ\pi_2\rangle)\circ\\ \mathbf{set}\end{array}\right\rangle\right)\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \texttt{K}_f\ (\texttt{P})\rangle)\circ\\ \mathbf{set}\end{array}$$

$$\equiv \begin{array}{l} \mathbf{iterate}\left(\texttt{K}_p\ (\texttt{T}),\ \left\langle\langle\texttt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l}\mathbf{iterate}\,(\mathbf{eq}\ \oplus\ \langle\texttt{addr}\circ\pi_2\circ\pi_1,\ \texttt{addr}\circ\pi_2\rangle,\ \pi_2)\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \texttt{child}\circ\pi_2\rangle)\circ\\ \mathbf{set}\end{array}\right\rangle\right)\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \texttt{K}_f\ (\texttt{P})\rangle)\circ\\ \mathbf{set}\end{array}$$

To demonstrate the correctness of the translation, we reduce the resulting KOLA function applied to the empty tuple, $\diamond$, according to the definitions of KOLA operators of Table 2 and the identities of Section 3.2.2.

$$\left(\begin{array}{l} \mathbf{iterate}\left(\texttt{K}_p\ (\texttt{T}),\ \left\langle\langle\texttt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l}\mathbf{iterate}\,(\mathbf{eq}\ \oplus\ \langle\texttt{addr}\circ\pi_2\circ\pi_1,\ \texttt{addr}\circ\pi_2\rangle,\ \pi_2)\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \texttt{child}\circ\pi_2\rangle)\circ\\ \mathbf{set}\end{array}\right\rangle\right)\circ\\ \mathbf{flat}\circ\\ \mathbf{iterate}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{iter}\,(\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \texttt{K}_f\ (\texttt{P})\rangle)\circ\\ \mathbf{set}\end{array}\right)\ !\ \diamond$$

$$= \begin{array}{l} \mathbf{iterate}\left(\mathtt{K}_p\ (\mathtt{T}),\ \left\langle \langle \mathtt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l} \mathbf{iterate}\ (\mathbf{eq}\ \oplus\ \langle \mathtt{addr} \circ \pi_2 \circ \pi_1,\ \mathtt{addr} \circ \pi_2\rangle,\ \pi_2) \circ \\ \mathbf{flat} \circ \\ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathtt{child} \circ \pi_2\rangle) \circ \\ \mathbf{set} \end{array} \right\rangle \right)\ ! \\ \left(\begin{array}{l} \mathbf{flat}\ ! \\ \left(\begin{array}{l} \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathtt{K}_f\ (\mathtt{P})\rangle)\ ! \\ (\mathbf{set}\ !\ \diamond) \end{array} \right) \end{array} \right) \end{array}$$

$$= \begin{array}{l} \mathbf{iterate}\left(\mathtt{K}_p\ (\mathtt{T}),\ \left\langle \langle \mathtt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l} \mathbf{iterate}\ (\mathbf{eq}\ \oplus\ \langle \mathtt{addr} \circ \pi_2 \circ \pi_1,\ \mathtt{addr} \circ \pi_2\rangle,\ \pi_2) \circ \\ \mathbf{flat} \circ \\ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathtt{child} \circ \pi_2\rangle) \circ \\ \mathbf{set} \end{array} \right\rangle \right)\ ! \\ \left(\mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct}\ (\mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathtt{K}_f\ (\mathtt{P})\rangle)\ \mathtt{!}\ V_0 \\ \mathtt{from\ set}\ (\diamond) \end{array} \right) \right) \end{array}$$

$$= \begin{array}{l} \mathbf{iterate}\left(\mathtt{K}_p\ (\mathtt{T}),\ \left\langle \langle \mathtt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l} \mathbf{iterate}\ (\mathbf{eq}\ \oplus\ \langle \mathtt{addr} \circ \pi_2 \circ \pi_1,\ \mathtt{addr} \circ \pi_2\rangle,\ \pi_2) \circ \\ \mathbf{flat} \circ \\ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathtt{child} \circ \pi_2\rangle) \circ \\ \mathbf{set} \end{array} \right\rangle \right)\ ! \\ \left(\mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct}\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\ \mathtt{!} \ll V_0,\ \mathtt{P} \gg \\ \mathtt{from\ set}\ (\diamond) \end{array} \right) \right) \end{array}$$

$$= \begin{array}{l} \mathbf{iterate}\left(\mathtt{K}_p\ (\mathtt{T}),\ \left\langle \langle \mathtt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l} \mathbf{iterate}\ (\mathbf{eq}\ \oplus\ \langle \mathtt{addr} \circ \pi_2 \circ \pi_1,\ \mathtt{addr} \circ \pi_2\rangle,\ \pi_2) \circ \\ \mathbf{flat} \circ \\ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathtt{child} \circ \pi_2\rangle) \circ \\ \mathbf{set} \end{array} \right\rangle \right)\ ! \\ \left(\mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct} \\ \\ \mathtt{from\ set}\ (\diamond) \end{array} \left(\begin{array}{l} \mathtt{select\ distinct}\ \mathbf{id}\ \mathtt{!} \ll V_0 \uparrow \binom{1}{0},\ V_0 \gg \\ \mathtt{from\ P} \end{array} \right) \right) \right) \end{array}$$

$$= \begin{array}{l} \mathbf{iterate}\left(\mathtt{K}_p\ (\mathtt{T}),\ \left\langle \langle \mathtt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l} \mathbf{iterate}\ (\mathbf{eq}\ \oplus\ \langle \mathtt{addr} \circ \pi_2 \circ \pi_1,\ \mathtt{addr} \circ \pi_2\rangle,\ \pi_2) \circ \\ \mathbf{flat} \circ \\ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathtt{child} \circ \pi_2\rangle) \circ \\ \mathbf{set} \end{array} \right\rangle \right)\ ! \\ \left(\mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct} \\ \\ \mathtt{from\ set}\ (\diamond) \end{array} \left(\begin{array}{l} \mathtt{select\ distinct} \ll V_1,\ V_0 \gg \\ \mathtt{from\ P} \end{array} \right) \right) \right) \end{array}$$

$$= \begin{array}{l} \mathbf{iterate}\left(\mathtt{K}_p\ (\mathtt{T}),\ \left\langle \langle \mathtt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l} \mathbf{iterate}\ (\mathbf{eq}\ \oplus\ \langle \mathtt{addr} \circ \pi_2 \circ \pi_1,\ \mathtt{addr} \circ \pi_2\rangle,\ \pi_2) \circ \\ \mathbf{flat} \circ \\ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathtt{child} \circ \pi_2\rangle) \circ \\ \mathbf{set} \end{array} \right\rangle \right)\ ! \\ \left(\begin{array}{l} \mathtt{select\ distinct} \ll V_1,\ V_0 \gg \\ \mathtt{from\ set}\ (\diamond),\ \mathtt{P} \end{array} \right) \end{array}$$

By Lemma 3.3

$$
= \begin{array}{l}
\mathbf{iterate}\left(\mathtt{K}_p\ (\mathtt{T}),\ \left\langle \langle \mathtt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l}
\mathbf{iterate}\,(\mathbf{eq}\ \oplus\ \langle \mathtt{addr}\circ\pi_2\circ\pi_1,\ \mathtt{addr}\circ\pi_2\rangle,\ \pi_2)\circ\\
\mathbf{flat}\ \circ\\
\ \mathbf{iterate}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \mathtt{child}\circ\pi_2\rangle)\circ\\
\ \ \mathbf{set}
\end{array} \right\rangle \right)\ \mathtt{!}\\
\left(\begin{array}{l}
\mathtt{select\ distinct} \ll V_1,\ V_0 \gg \downarrow \binom{\diamond\,\uparrow\,\binom{1}{0}}{1}\\
\mathtt{from\ P} \downarrow \binom{\diamond}{0}
\end{array} \right)
\end{array}
$$

By Lemma 3.6

$$
= \begin{array}{l}
\mathbf{iterate}\left(\mathtt{K}_p\ (\mathtt{T}),\ \left\langle \langle \mathtt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l}
\mathbf{iterate}\,(\mathbf{eq}\ \oplus\ \langle \mathtt{addr}\circ\pi_2\circ\pi_1,\ \mathtt{addr}\circ\pi_2\rangle,\ \pi_2)\circ\\
\mathbf{flat}\ \circ\\
\ \mathbf{iterate}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \mathtt{child}\circ\pi_2\rangle)\circ\\
\ \ \mathbf{set}
\end{array} \right\rangle \right)\ \mathtt{!}\\
\left(\begin{array}{l}
\mathtt{select\ distinct} \ll \diamond,\ V_0 \gg\\
\mathtt{from\ P}
\end{array} \right)
\end{array}
$$

$$
= \begin{array}{l}
\mathtt{select\ distinct}\ \left\langle \langle \mathtt{K}_f\ (\diamond),\ \pi_2\rangle,\ \begin{array}{l}
\mathbf{iterate}\,(\mathbf{eq}\ \oplus\ \langle \mathtt{addr}\circ\pi_2\circ\pi_1,\ \mathtt{addr}\circ\pi_2\rangle,\ \pi_2)\circ\\
\mathbf{flat}\ \circ\\
\ \mathbf{iterate}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \mathtt{child}\circ\pi_2\rangle)\circ\\
\ \ \mathbf{set}
\end{array} \right\rangle\ \mathtt{!}\ V_0\\
\mathtt{from}\ \left(\begin{array}{l}
\mathtt{select\ distinct} \ll \diamond,\ V_0 \gg\\
\mathtt{from\ P}
\end{array} \right)
\end{array}
$$

$$
= \begin{array}{l}
\mathtt{select\ distinct}\\
\quad \ll \diamond,\ \pi_2\ \mathtt{!}\ V_0,\ \begin{array}{l}
\mathbf{iterate}\,(\mathbf{eq}\ \oplus\ \langle \mathtt{addr}\circ\pi_2\circ\pi_1,\ \mathtt{addr}\circ\pi_2\rangle,\ \pi_2)\circ\\
\mathbf{flat}\ \circ\\
\ \mathbf{iterate}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \mathtt{child}\circ\pi_2\rangle)\circ\\
\ \ \mathbf{set}
\end{array}\ \mathtt{!}\ V_0 \gg\\
\mathtt{from}\ \left(\begin{array}{l}
\mathtt{select\ distinct} \ll \diamond,\ V_0 \gg\\
\mathtt{from\ P}
\end{array} \right)
\end{array}
$$

$$
= \begin{array}{l}
\mathtt{select\ distinct}\\
\quad \ll \diamond,\ \pi_2\ \mathtt{!}\ V_0,\\
\qquad \left(\begin{array}{l}
\mathtt{select\ distinct}\ \pi_2\ \mathtt{!}\ V_0\\
\mathtt{from}\ \left(\mathtt{flatten}\ \left(\begin{array}{l}
\mathtt{select\ distinct}\\
\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\circ\langle\mathbf{id},\ \mathtt{child}\circ\pi_2\rangle\ \mathtt{!}\ V_0\\
\mathtt{from\ set}\ (V_0)
\end{array} \right) \right)\\
\mathtt{where}\ \mathbf{eq}\ \oplus\ \langle \mathtt{addr}\circ\pi_2\circ\pi_1,\ \mathtt{addr}\circ\pi_2\rangle\ \mathtt{?}\ V_0
\end{array} \right) \gg\\
\mathtt{from}\ \left(\begin{array}{l}
\mathtt{select\ distinct} \ll \diamond,\ V_0 \gg\\
\mathtt{from\ P}
\end{array} \right)
\end{array}
$$

$$
\begin{array}{ll}
 & \texttt{select distinct} \\
 & \quad \ll \diamond,\ \pi_2\ !\ V_0, \\
= & \qquad \left(\begin{array}{l} \texttt{select distinct } \pi_2\ !\ V_0 \\ \texttt{from} \left(\texttt{flatten} \left(\begin{array}{l} \texttt{select distinct} \\ \ \mathbf{iter}\ (\texttt{K}_p\ (\texttt{T}),\ \mathbf{id})\ !\ \begin{array}{l} \ll V_0, \\ \quad (\texttt{child} \circ \pi_2)\ !\ V_0 \gg \end{array} \\ \texttt{from set } (V_0) \end{array} \right) \right) \gg \\ \texttt{where } \mathbf{eq} \oplus \langle \texttt{addr} \circ \pi_2 \circ \pi_1,\ \texttt{addr} \circ \pi_2 \rangle\ ?\ V_0 \end{array} \right) \gg \\
 & \texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array} \right)
\end{array}
$$

$$
\begin{array}{ll}
 & \texttt{select distinct} \\
= & \quad \ll \diamond,\ \pi_2\ !\ V_0, \left(\begin{array}{l} \texttt{select distinct } \pi_2\ !\ V_0 \\ \texttt{from} \left(\texttt{flatten} \left(\begin{array}{l} \texttt{select distinct} \\ \ (\texttt{select distinct} \ll V_0 \uparrow \binom{1}{0},\ V_0 \gg \\ \quad \texttt{from } (\texttt{child} \circ \pi_2)\ !\ V_0) \\ \texttt{from set } (V_0) \end{array} \right) \right) \\ \texttt{where } \mathbf{eq} \oplus \langle \texttt{addr} \circ \pi_2 \circ \pi_1,\ \texttt{addr} \circ \pi_2 \rangle\ ?\ V_0 \end{array} \right) \gg \\
 & \texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array} \right)
\end{array}
$$

$$
\begin{array}{ll}
 & \texttt{select distinct} \\
= & \quad \ll \diamond,\ \pi_2\ !\ V_0, \left(\begin{array}{l} \texttt{select distinct } \pi_2\ !\ V_0 \\ \texttt{from} \left(\texttt{flatten} \left(\begin{array}{l} \texttt{select distinct} \\ \ (\texttt{select distinct} \ll V_1,\ V_0 \gg \\ \quad \texttt{from } (\texttt{child} \circ \pi_2)\ !\ V_0) \\ \texttt{from set } (V_0) \end{array} \right) \right) \\ \texttt{where } \mathbf{eq} \oplus \langle \texttt{addr} \circ \pi_2 \circ \pi_1,\ \texttt{addr} \circ \pi_2 \rangle\ ?\ V_0 \end{array} \right) \gg \\
 & \texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array} \right)
\end{array}
$$

$$
\begin{array}{ll}
 & \texttt{select distinct} \\
= & \quad \ll \diamond,\ \pi_2\ !\ V_0, \left(\begin{array}{l} \texttt{select distinct } \pi_2\ !\ V_0 \\ \texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll V_1,\ V_0 \gg \\ \texttt{from set } (V_0),\ (\texttt{child} \circ \pi_2)\ !\ V_0 \end{array} \right) \\ \texttt{where } \mathbf{eq} \oplus \langle \texttt{addr} \circ \pi_2 \circ \pi_1,\ \texttt{addr} \circ \pi_2 \rangle\ ?\ V_0 \end{array} \right) \gg \\
 & \texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array} \right)
\end{array}
$$

(By Lemma 3.3)

$$
= \quad \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0 \,.\, a_1,\ \left(\begin{array}{l}
\texttt{select distinct}\ V_0 \,.\, a_1 \\
\texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll V_1,\ V_0 \gg \\ \texttt{from set}\ (V_0),\ ((V_0 \,.\, a_1) \rightarrow \texttt{child}) \end{array} \right) \\
\texttt{where}\ ((V_0 \,.\, a_0 \,.\, a_1) \rightarrow \texttt{addr})\ \texttt{==}\ ((V_0 \,.\, a_1) \rightarrow \texttt{addr})
\end{array} \right) \gg \\
\texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array} \right)
\end{array}
$$

$$
= \quad \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0 \,.\, a_1,\ \left(\begin{array}{l}
\texttt{select distinct}\ ((V_0 \,.\, a_1) \uparrow \tbinom{2}{1}) \downarrow \tbinom{\ll V_1,\ V_0 \gg}{0} \\
\texttt{from set}\ (V_0),\ ((V_0 \,.\, a_1) \rightarrow \texttt{child}) \\
\texttt{where}\ ((((V_0 \,.\, a_0 \,.\, a_1) \rightarrow \texttt{addr})\ \texttt{==}\ ((V_0 \,.\, a_1) \rightarrow \texttt{addr})) \uparrow \tbinom{2}{1}) \downarrow \tbinom{\ll V_1,\ V_0 \gg}{0}
\end{array} \right) \gg \\
\texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array} \right)
\end{array}
$$

(By Lemma 3.8)

$$
= \quad \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0 \,.\, a_1, \\
\qquad \left(\begin{array}{l}
\texttt{select distinct} \ll V_1,\ V_0 \gg .\, a_1 \\
\texttt{from set}\ (V_0),\ ((V_0 \,.\, a_1) \rightarrow \texttt{child}) \\
\texttt{where}\ (((V_0 \,.\, a_0 \,.\, a_1) \uparrow \tbinom{2}{1} \downarrow \tbinom{\ll V_1,\ V_0 \gg}{0})) \rightarrow \texttt{addr})\ \texttt{==}\ (((V_0 \,.\, a_1) \uparrow \tbinom{2}{1} \downarrow \tbinom{\ll V_1,\ V_0 \gg}{0})) \rightarrow \texttt{addr})
\end{array} \right) \gg \\
\texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array} \right)
\end{array}
$$

$$
= \quad \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0 \,.\, a_1,\ \left(\begin{array}{l}
\texttt{select distinct}\ V_0 \\
\texttt{from set}\ (V_0),\ ((V_0 \,.\, a_1) \rightarrow \texttt{child}) \\
\texttt{where}\ ((\ll V_1,\ V_0 \gg .\, a_0 \,.\, a_1) \rightarrow \texttt{addr})\ \texttt{==}\ ((\ll V_1,\ V_0 \gg .\, a_1) \rightarrow \texttt{addr})
\end{array} \right) \gg \\
\texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array} \right)
\end{array}
$$

$$
= \quad \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0 \,.\, a_1,\ \left(\begin{array}{l}
\texttt{select distinct}\ V_0 \\
\texttt{from set}\ (V_0),\ ((V_0 \,.\, a_1) \rightarrow \texttt{child}) \\
\texttt{where}\ ((V_1 \,.\, a_1) \rightarrow \texttt{addr})\ \texttt{==}\ (V_0 \rightarrow \texttt{addr})
\end{array} \right) \gg \\
\texttt{from} \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array} \right)
\end{array}
$$

$$
= \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0\ .\ a_1,\ \left(\begin{array}{l} \texttt{select distinct } V_0 \downarrow \binom{V_0 \uparrow \binom{1}{0}}{1} \\ \texttt{from } (((V_0\ .\ a_1) \rightarrow \texttt{child})) \downarrow \binom{V_0}{0} \\ \texttt{where } (((V_1\ .\ a_1) \rightarrow \texttt{addr})\ ==\ (V_0 \rightarrow \texttt{addr})) \downarrow \binom{V_0 \uparrow \binom{1}{0}}{1} \end{array}\right) \gg \\
\texttt{from } \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array}\right)
\end{array}
$$

(By Lemma 3.6)

$$
= \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0\ .\ a_1,\ \left(\begin{array}{l} \texttt{select distinct } V_0 \\ \texttt{from } ((V_0\ .\ a_1) \rightarrow \texttt{child}) \\ \texttt{where } ((V_1\ .\ a_1) \rightarrow \texttt{addr})\ ==\ (V_0 \rightarrow \texttt{addr}) \end{array}\right) \gg \\
\texttt{from } \left(\begin{array}{l} \texttt{select distinct} \ll \diamond,\ V_0 \gg \\ \texttt{from P} \end{array}\right)
\end{array}
$$

$$
= \begin{array}{l}
\texttt{select distinct} \\
\quad \ll (\diamond \uparrow \binom{1}{1}) \downarrow \binom{\ll \diamond,\ V_0 \gg}{0},\ ((V_0\ .\ a_1) \uparrow \binom{1}{1}) \downarrow \binom{\ll \diamond,\ V_0 \gg}{0}, \\
\qquad \left(\left(\begin{array}{l} \texttt{select distinct } V_0 \\ \texttt{from } ((V_0\ .\ a_1) \rightarrow \texttt{child}) \\ \texttt{where } ((V_1\ .\ a_1) \rightarrow \texttt{addr})\ ==\ (V_0 \rightarrow \texttt{addr}) \end{array}\right) \uparrow \binom{1}{1} \right) \downarrow \binom{\ll \diamond,\ V_0 \gg}{0} \gg \\
\texttt{from P}
\end{array}
$$

(By Lemma 3.8)

$$
= \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0,\ \left(\begin{array}{l} \texttt{select distinct } V_0 \uparrow \binom{1}{2} \\ \texttt{from } ((V_0\ .\ a_1) \rightarrow \texttt{child}) \uparrow \binom{1}{1} \\ \texttt{where } (((V_1\ .\ a_1) \rightarrow \texttt{addr})\ ==\ (V_0 \rightarrow \texttt{addr})) \uparrow \binom{1}{2} \end{array}\right) \downarrow \binom{\ll \diamond,\ V_0 \gg}{0} \gg \\
\texttt{from P}
\end{array}
$$

$$
= \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0,\ \left(\begin{array}{l} \texttt{select distinct } V_0 \\ \texttt{from } ((V_0\ .\ a_1) \rightarrow \texttt{child}) \\ \texttt{where } ((V_1\ .\ a_1) \rightarrow \texttt{addr})\ ==\ (V_0 \rightarrow \texttt{addr}) \end{array}\right) \downarrow \binom{\ll \diamond,\ V_0 \gg}{0} \gg \\
\texttt{from P}
\end{array}
$$

$$
= \begin{array}{l}
\texttt{select distinct} \\
\quad \ll \diamond,\ V_0,\ \left(\begin{array}{l} \texttt{select distinct } V_0 \downarrow \binom{\ll \diamond,\ V_0 \gg \uparrow \binom{1}{0}}{1} \\ \texttt{from } ((V_0\ .\ a_1) \rightarrow \texttt{child}) \downarrow \binom{\ll \diamond,\ V_0 \gg}{0} \\ \texttt{where } (((V_1\ .\ a_1) \rightarrow \texttt{addr})\ ==\ (V_0 \rightarrow \texttt{addr})) \downarrow \binom{\ll \diamond,\ V_0 \gg \uparrow \binom{1}{0}}{1} \end{array}\right) \gg \\
\texttt{from P}
\end{array}
$$

$$= \begin{array}{l} \texttt{select distinct} \\ \quad \ll \diamond,\ V_0,\ \left(\begin{array}{l} \texttt{select distinct } V_0 \\ \texttt{from } ((\ll \diamond,\ V_0 \gg . \ a_1) \rightarrow \texttt{child}) \\ \texttt{where } ((\ll \diamond,\ V_1 \gg . \ a_1) \rightarrow \texttt{addr}) \ \ \texttt{==} \ \ (V_0 \rightarrow \texttt{addr}) \end{array} \right) \gg \\ \texttt{from P} \end{array}$$

$$= \begin{array}{l} \texttt{select distinct} \\ \quad \ll \diamond,\ V_0,\ \left(\begin{array}{l} \texttt{select distinct } V_0 \\ \texttt{from } (V_0 \rightarrow \texttt{child}) \\ \texttt{where } (V_1 \rightarrow \texttt{addr}) \ \ \texttt{==} \ \ (V_0 \rightarrow \texttt{addr}) \end{array} \right) \gg \\ \texttt{from P} \end{array}$$

4.3 The Correctness of Translation

In this section, we present proofs establishing the correctness of the translation function $\mathbf{T}\ [\![]\!]$.

4.4 Translator Correctness

The correctness of translation is established by showing for any *wfq* Q, that the invocation of $\mathbf{T}\ [\![Q]\!]$ on $\diamond$ reduces to an expression with the same semantics as Q. (i.e,

$$\begin{array}{lrcl} 1. & \mathbf{Eval}\ [\![Q]\!]\ \epsilon & = & \mathbf{Eval}\ [\![\mathbf{T}\ [\![Q]\!]\ !\ \diamond]\!]\ \epsilon, \text{ and} \\ 2. & \mathbf{Eval}\ [\![Q]\!]\ \epsilon & = & \mathbf{Eval}\ [\![\mathbf{T}\ [\![Q]\!]\ ?\ \diamond]\!]\ \epsilon \end{array}$$

depending on whether Q evaluates to a non-Boolean value (1) or a Boolean value (2). Below (**Theorem 4.1**) we show the stronger result that for any OQL$_{deB}$ expressions Q and Q' and environment ρ such that Q' is well-formed with respect to ρ and Q is well-formed with respect to $\mathbf{Eval}\ [\![Q']\!]\ \rho$,

$$\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q]\!]\ !\ Q']\!]\ \rho \ = \ \mathbf{Eval}\ [\![Q]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho).$$
for any non-Boolean expression, Q, and

$$\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q]\!]\ ?\ Q']\!]\ \rho \ = \ \mathbf{Eval}\ [\![Q]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho).$$
for any Boolean expression, Q.

Our desired result follows by setting Q' to $\diamond$ and ρ to ϵ, as this gives:

4.4.1 Initial Lemmas

We begin by proving **Lemma 4.1** that shows that the translation of variables is correct.

Lemma 4.1 *Given variable expression V_i, OQL$_{deB}$ expression Q' and environment ρ such that Q' is well-formed with respect to ρ and V_i is well-formed with respect to* $\mathbf{Eval}\ [\![Q']\!]\ \rho$,

$$\mathbf{Eval}\ [\![\mathbf{T}\ [\![V_i]\!]\ !\ Q']\!]\ \rho \ = \ \mathbf{Eval}\ [\![V_i]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho).$$

Proof: Note that since V_i is well-formed with respect to $\mathbf{Eval}\ [\![Q']\!]\ \rho$, we have

$$\mathbf{Eval}\ [\![Q']\!]\ \rho \ = \ (\delta,\ v_0)$$

for some $\delta = (\epsilon,\ v_m, \ldots,\ v_i, \ldots,\ v_1) \in \mathcal{D}$ and $(0 \leq i \leq m)$. (If $m = 0$, then $\delta = \epsilon$.) We proceed with a proof by induction on i. For the base case, $i = 0$ and:

$$\begin{aligned}
\textbf{Eval}\ [\![\textbf{T}\ [\![V_0]\!]\ !\ Q']\!]\ \rho &= \textbf{Eval}\ [\![\pi_2\ !\ Q']\!]\ \rho \\
&= \textbf{Eval}\ [\![Q'\ .\ a_1]\!]\ \rho \\
&= v_0,\ \text{ if } \textbf{Eval}\ [\![Q']\!]\ \rho\ =\ (\delta,\ v_0) \\
&= \textbf{Eval}\ [\![V_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}$$

For the inductive case, we assume that the lemma holds for all values of $i < k$ (some $k > 0$) and show that the lemma holds when $i = k$:

$$\begin{aligned}
&\textbf{Eval}\ [\![\textbf{T}\ [\![V_k]\!]\ !\ Q']\!]\ \rho \\
&\quad = \textbf{Eval}\ \left[\!\!\left[(\pi_2\ \circ\ {\pi_1}^k)\ !\ Q'\right]\!\!\right]\ \rho \\
&\quad = \textbf{Eval}\ \left[\!\!\left[(\pi_2\ \circ\ {\pi_1}^{k-1}\ \circ\ \pi_1)\ !\ Q'\right]\!\!\right]\ \rho \\
&\quad = \textbf{Eval}\ \left[\!\!\left[(\pi_2\ \circ\ {\pi_1}^{k-1})\ !\ (\pi_1\ !\ Q')\right]\!\!\right]\ \rho \\
&\quad = \textbf{Eval}\ [\![\textbf{T}\ [\![V_{k-1}]\!]\ !\ (Q'\ .\ a_0)]\!]\ \rho \\
&\quad = \textbf{Eval}\ [\![V_{k-1}]\!]\ (\textbf{Eval}\ [\![Q'\ .\ a_0]\!]\ \rho)\ \ \text{(by induction)} \\
&\quad = \textbf{Eval}\ [\![V_{k-1}]\!]\ (\epsilon,\ v_m, \ldots,\ v_k,\ v_{k-1}, \ldots,\ v_1)\ \text{ if } \textbf{Eval}\ [\![Q']\!]\ \rho\ =\ (\epsilon,\ v_m, \ldots,\ v_k,\ v_{k-1}, \ldots,\ v_0) \\
&\quad = v_k \\
&\quad = \textbf{Eval}\ [\![V_k]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)\ \ \square
\end{aligned}$$

Next, we prove a result that will be of use when we show the correctness of translation of `select−distinct` queries in Theorem 4.1. This lemma establishes the equivalence of a complex KOLA function produced as a result of translation and the OQL$_{deB}$ expression to which it is equivalent.

Lemma 4.2 *For any KOLA functions $f_0,\ \ldots,\ f_m$, OQL$_{deB}$ expression Q, and $m \geq 0$,*

$$\begin{array}{l}
(\textbf{flat}\ \circ\ \textbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \textbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \textbf{id})\ \circ\ \langle\textbf{id},\ f_m\rangle)\ \circ \\
\quad \ldots \\
\quad \textbf{flat}\ \circ\ \textbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \textbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \textbf{id})\ \circ\ \langle\textbf{id},\ f_0\rangle)\ \circ \\
\quad \textbf{set})\ !\ Q \\
\qquad\qquad = \\
\mathtt{select\ distinct} \ll V_{m+1}, \ldots,\ V_0 \gg \\
\mathtt{from} \left(\begin{array}{l}
\mathtt{set}\ (Q), \\
f_0\ !\ V_0, \\
((f_1\ !\ V_0) \uparrow \binom{2}{1}) \downarrow \binom{\ll V_1, \ldots,\ V_0 \gg}{0}, \ldots, \\
((f_m\ !\ V_0) \uparrow \binom{m+1}{1}) \downarrow \binom{\ll V_m, \ldots,\ V_0 \gg}{0}
\end{array} \right)
\end{array}$$

Proof: The proof is by induction on m. For the base cases we have

- $m = 0$:

$$(\textbf{flat}\ \circ\ \textbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \textbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \textbf{id})\ \circ\ \langle\textbf{id},\ f_0\rangle)\ \circ\ \textbf{set})\ !\ Q$$

$$= \quad \mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct}\ (\textbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \textbf{id})\ \circ\ \langle\textbf{id},\ f_0\rangle)\ !\ V_0 \\ \mathtt{from\ set}\ (Q) \end{array} \right)$$

$$= \quad \mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct} \left(\begin{array}{l} \mathtt{select\ distinct} \ll V_0 \uparrow \binom{1}{0},\ V_0 \gg \\ \mathtt{from}\ f_0\ \mathtt{!}\ V_0 \end{array} \right) \\ \mathtt{from\ set}\ (Q) \end{array} \right)$$

$$= \quad \mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct} \left(\begin{array}{l} \mathtt{select\ distinct} \ll V_1,\ V_0 \gg \\ \mathtt{from}\ f_0\ \mathtt{!}\ V_0 \end{array} \right) \\ \mathtt{from\ set}\ (Q) \end{array} \right)$$

$$= \quad \begin{array}{l} \mathtt{select\ distinct} \ll V_1,\ V_0 \gg \\ \mathtt{from\ set}\ (Q),\ f_0\ \mathtt{!}\ V_0 \end{array} \quad \text{(By Lemma 3.4)}$$

- and $m = 1$:

$$\left(\begin{array}{l} \mathbf{flat} \circ \mathbf{iterate}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ f_1 \rangle) \circ \\ \ \mathbf{flat} \circ \mathbf{iterate}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ f_0 \rangle) \circ \mathbf{set} \end{array} \right)\ \mathtt{!}\ Q$$

$$= \quad \mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct}\ (\mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ f_1 \rangle)\ \mathtt{!}\ V_0 \\ \mathtt{from} \left(\mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct}\ (\mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ f_0 \rangle)\ \mathtt{!}\ V_0 \\ \mathtt{from\ set}\ (Q) \end{array} \right) \right) \end{array} \right)$$

$$= \quad \mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct} \left(\begin{array}{l} \mathtt{select\ distinct} \ll V_0 \uparrow \binom{1}{0},\ V_0 \gg \\ \mathtt{from}\ f_1\ \mathtt{!}\ V_0 \end{array} \right) \\ \mathtt{from} \left(\mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct}\ (\mathtt{select\ distinct} \ll V_0 \uparrow \binom{1}{0},\ V_0 \gg \\ \qquad\qquad\qquad \mathtt{from}\ f_0\ \mathtt{!}\ V_0) \\ \mathtt{from\ set}\ (Q) \end{array} \right) \right) \end{array} \right)$$

$$= \quad \mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct} \left(\begin{array}{l} \mathtt{select\ distinct} \ll V_1,\ V_0 \gg \\ \mathtt{from}\ f_1\ \mathtt{!}\ V_0 \end{array} \right) \\ \mathtt{from} \left(\mathtt{flatten} \left(\begin{array}{l} \mathtt{select\ distinct}\ (\mathtt{select\ distinct} \ll V_1,\ V_0 \gg \\ \qquad\qquad\qquad \mathtt{from}\ f_0\ \mathtt{!}\ V_0) \\ \mathtt{from\ set}\ (Q) \end{array} \right) \right) \end{array} \right)$$

$$= \quad \begin{array}{l} \mathtt{select\ distinct} \ll V_1,\ V_0 \gg \\ \mathtt{from} \left(\begin{array}{l} \mathtt{select\ distinct} \ll V_1,\ V_0 \gg \\ \mathtt{from\ set}\ (Q),\ f_0\ \mathtt{!}\ V_0 \end{array} \right),\ f_1\ \mathtt{!}\ V_0 \end{array} \quad \text{(By Lemma 3.3)}$$

$$= \quad \begin{array}{l} \mathtt{select\ distinct}\ (\ll V_1,\ V_0 \gg \uparrow \binom{2}{2}) \downarrow \binom{\ll V_1,\ V_0 \gg \uparrow \binom{1}{0}}{1} \\ \mathtt{from\ set}\ (Q),\ f_0\ \mathtt{!}\ V_0,\ ((f_1\ \mathtt{!}\ V_0) \uparrow \binom{2}{1}) \downarrow \binom{\ll V_1,\ V_0 \gg}{0} \end{array} \quad \text{(By Lemma 3.7)}$$

$$= \quad \begin{array}{l} \mathtt{select\ distinct} \ll V_1,\ V_0 \gg \downarrow \binom{\ll V_2,\ V_1 \gg}{1} \\ \mathtt{from\ set}\ (Q),\ f_0\ \mathtt{!}\ V_0,\ ((f_1\ \mathtt{!}\ V_0) \uparrow \binom{2}{1}) \downarrow \binom{\ll V_1,\ V_0 \gg}{0} \end{array}$$

$$= \quad \begin{array}{l} \mathtt{select\ distinct} \ll \ll V_2,\ V_1 \gg,\ V_0 \gg \\ \mathtt{from\ set}\ (Q),\ f_0\ \mathtt{!}\ V_0,\ ((f_1\ \mathtt{!}\ V_0) \uparrow \binom{2}{1}) \downarrow \binom{\ll V_1,\ V_0 \gg}{0} \end{array}$$

$$= \quad \begin{array}{l} \mathtt{select\ distinct} \ll V_2, \ldots,\ V_0 \gg \\ \mathtt{from\ set}\ (Q),\ f_0\ \mathtt{!}\ V_0,\ ((f_1\ \mathtt{!}\ V_0) \uparrow \binom{2}{1}) \downarrow \binom{\ll V_1,\ V_0 \gg}{0} \end{array}$$

For the inductive case, we assume that the lemma holds for all values of m such that $0 \leq m < n$ (some $n > 1$) and show that the lemma holds for $m = n$:

$$\left(\begin{array}{l} \mathbf{flat} \circ \mathbf{iterate}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ f_n \rangle) \circ \\ \ \mathbf{flat} \circ \mathbf{iterate}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ f_{n-1} \rangle) \circ \\ \quad \ldots \\ \quad \mathbf{flat} \circ \mathbf{iterate}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\,(\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ f_0 \rangle) \circ \\ \quad\ \mathbf{set} \end{array} \right)\ \mathtt{!}\ Q$$

$$
= \quad \mathbf{flat}\ \mathtt{!}\ \left(\begin{array}{l}
\mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\ \circ\ \langle \mathbf{id},\ f_n \rangle)\ \mathtt{!} \\
\left(\left(\begin{array}{l}
\mathbf{flat}\ \circ\ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\ \circ\ \langle \mathbf{id},\ f_{n-1} \rangle)\ \circ \\
\ldots \\
\mathbf{flat}\ \circ\ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\ \circ\ \langle \mathbf{id},\ f_0 \rangle)\ \circ \\
\mathbf{set}
\end{array} \right)\ \mathtt{!}\ Q \right)
\end{array} \right)
$$

$$
= \quad \mathtt{flatten} \left(\begin{array}{l}
\mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id})\ \circ\ \langle \mathbf{id},\ f_n \rangle)\ \mathtt{!} \\
\left(\begin{array}{ll}
\multicolumn{2}{l}{\mathtt{select\ distinct} \ll V_n, \ldots,\ V_0 \gg} \\
\mathtt{from} & \left(\begin{array}{l}
\mathtt{set}\ (Q), \\
f_0\ \mathtt{!}\ V_0, \\
((f_1\ \mathtt{!}\ V_0) \uparrow \binom{2}{1}) \downarrow \binom{\ll V_1, \ldots,\ V_0 \gg}{0}, \ldots, \\
((f_{n-1}\ \mathtt{!}\ V_0) \uparrow \binom{n}{1}) \downarrow \binom{\ll V_{n-1}, \ldots,\ V_0 \gg}{0}
\end{array} \right)
\end{array} \right)
\end{array} \right)
$$

(by induction)

$$
= \quad \mathtt{flatten} \left(\begin{array}{ll}
\mathtt{select\ distinct} & \left(\begin{array}{l}
\mathtt{select\ distinct}\ \ll V_0 \uparrow \binom{1}{0},\ V_0 \gg \\
\mathtt{from}\ f_n\ \mathtt{!}\ V_0
\end{array} \right) \\
\mathtt{from} & \left(\begin{array}{ll}
\multicolumn{2}{l}{\mathtt{select\ distinct} \ll V_n, \ldots,\ V_0 \gg} \\
\mathtt{from} & \left(\begin{array}{l}
\mathtt{set}\ (Q), \\
f_0\ \mathtt{!}\ V_0, \\
((f_1\ \mathtt{!}\ V_0) \uparrow \binom{2}{1}) \downarrow \binom{\ll V_1, \ldots,\ V_0 \gg}{0}, \ldots, \\
((f_{n-1}\ \mathtt{!}\ V_0) \uparrow \binom{n}{1}) \downarrow \binom{\ll V_{n-1}, \ldots,\ V_0 \gg}{0}
\end{array} \right)
\end{array} \right)
\end{array} \right)
$$

$$
= \quad \mathtt{flatten} \left(\begin{array}{ll}
\mathtt{select\ distinct} & \left(\begin{array}{l}
\mathtt{select\ distinct}\ \ll V_1,\ V_0 \gg \\
\mathtt{from}\ f_n\ \mathtt{!}\ V_0
\end{array} \right) \\
\mathtt{from} & \left(\begin{array}{ll}
\multicolumn{2}{l}{\mathtt{select\ distinct} \ll V_n, \ldots,\ V_0 \gg} \\
\mathtt{from} & \left(\begin{array}{l}
\mathtt{set}\ (Q), \\
f_0\ \mathtt{!}\ V_0, \\
((f_1\ \mathtt{!}\ V_0) \uparrow \binom{2}{1}) \downarrow \binom{\ll V_1, \ldots,\ V_0 \gg}{0}, \ldots, \\
((f_{n-1}\ \mathtt{!}\ V_0) \uparrow \binom{n}{1}) \downarrow \binom{\ll V_{n-1}, \ldots,\ V_0 \gg}{0}
\end{array} \right)
\end{array} \right)
\end{array} \right)
$$

$$
= \quad \begin{array}{ll}
\multicolumn{2}{l}{\mathtt{select\ distinct}\ \ll V_1,\ V_0 \gg} \\
\mathtt{from} & \left(\left(\begin{array}{ll}
\multicolumn{2}{l}{\mathtt{select\ distinct} \ll V_n, \ldots,\ V_0 \gg} \\
\mathtt{from} & \left(\begin{array}{l}
\mathtt{set}\ (Q), \\
f_0\ \mathtt{!}\ V_0, \\
((f_1\ \mathtt{!}\ V_0) \uparrow \binom{2}{1}) \downarrow \binom{\ll V_1, \ldots,\ V_0 \gg}{0}, \ldots, \\
((f_{n-1}\ \mathtt{!}\ V_0) \uparrow \binom{n}{1}) \downarrow \binom{\ll V_{n-1}, \ldots,\ V_0 \gg}{0}
\end{array} \right)
\end{array} \right),\ f_n\ \mathtt{!}\ V_0 \right)
\end{array}
$$

(By Lemma 3.3)

$$
= \begin{array}{l}
\texttt{select distinct } (\ll V_1,\ V_0 \gg \uparrow ({}^{n+1}_{2})) \downarrow ({}^{\ll V_n, \ldots,\ V_0 \gg \uparrow ({}^{1}_{0})}_{1}) \\
\texttt{from} \left(\begin{array}{l}
\texttt{set } (Q), \\
f_0 \texttt{ ! } V_0, \\
((f_1 \texttt{ ! } V_0) \uparrow ({}^{2}_{1})) \downarrow ({}^{\ll V_1, \ldots,\ V_0 \gg}_{0}), \ldots, \\
((f_{n-1} \texttt{ ! } V_0) \uparrow ({}^{n}_{1})) \downarrow ({}^{\ll V_{n-1}, \ldots,\ V_0 \gg}_{0}), \\
((f_n \texttt{ ! } V_0) \uparrow ({}^{n+1}_{1})) \downarrow ({}^{\ll V_n, \ldots,\ V_0 \gg}_{0})
\end{array} \right)
\end{array}
$$

(By Lemma 3.7)

$$
= \begin{array}{l}
\texttt{select distinct } \ll V_1,\ V_0 \gg \downarrow ({}^{\ll V_{n+1}, \ldots,\ V_1 \gg}_{1}) \\
\texttt{from} \left(\begin{array}{l}
\texttt{set } (Q), \\
f_0 \texttt{ ! } V_0, \\
((f_1 \texttt{ ! } V_0) \uparrow ({}^{2}_{1})) \downarrow ({}^{\ll V_1, \ldots,\ V_0 \gg}_{0}), \ldots, \\
((f_{n-1} \texttt{ ! } V_0) \uparrow ({}^{n}_{1})) \downarrow ({}^{\ll V_{n-1}, \ldots,\ V_0 \gg}_{0}), \\
((f_n \texttt{ ! } V_0) \uparrow ({}^{n+1}_{1})) \downarrow ({}^{\ll V_n, \ldots,\ V_0 \gg}_{0})
\end{array} \right)
\end{array}
$$

$$
= \begin{array}{l}
\texttt{select distinct } \ll \ll V_{n+1}, \ldots,\ V_1 \gg,\ V_0 \gg \\
\texttt{from} \left(\begin{array}{l}
\texttt{set } (Q), \\
f_0 \texttt{ ! } V_0, \\
((f_1 \texttt{ ! } V_0) \uparrow ({}^{2}_{1})) \downarrow ({}^{\ll V_1, \ldots,\ V_0 \gg}_{0}), \ldots, \\
((f_{n-1} \texttt{ ! } V_0) \uparrow ({}^{n}_{1})) \downarrow ({}^{\ll V_{n-1}, \ldots,\ V_0 \gg}_{0}), \\
((f_n \texttt{ ! } V_0) \uparrow ({}^{n+1}_{1})) \downarrow ({}^{\ll V_n, \ldots,\ V_0 \gg}_{0})
\end{array} \right)
\end{array}
$$

$$
= \begin{array}{l}
\texttt{select distinct } \ll V_{n+1}, \ldots,\ V_0 \gg \\
\texttt{from} \left(\begin{array}{l}
\texttt{set } (Q), \\
f_0 \texttt{ ! } V_0, \\
((f_1 \texttt{ ! } V_0) \uparrow ({}^{2}_{1})) \downarrow ({}^{\ll V_1, \ldots,\ V_0 \gg}_{0}), \ldots, \\
((f_{n-1} \texttt{ ! } V_0) \uparrow ({}^{n}_{1})) \downarrow ({}^{\ll V_{n-1}, \ldots,\ V_0 \gg}_{0}), \\
((f_n \texttt{ ! } V_0) \uparrow ({}^{n+1}_{1})) \downarrow ({}^{\ll V_n, \ldots,\ V_0 \gg}_{0})
\end{array} \right)
\end{array} \qquad \square
$$

4.4.2 Translator Correctness

The strong correctness result is proven by structural induction on any OQL_{deB} expression Q.

Theorem 4.1 (Correctness) *For any environment ρ, non-Boolean OQL_{deB} expression Q' that is well-formed with respect to ρ, and OQL_{deB} expression Q that is well-formed with respect to* **Eval** $[\![Q']\!]\ \rho$*,*

1. $\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q]\!] \texttt{ ! } Q']\!]\ \rho\ =\ \mathbf{Eval}\ [\![Q]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)$
   if Q is non-Boolean, and

2. $\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q]\!] \texttt{ ? } Q']\!]\ \rho\ =\ \mathbf{Eval}\ [\![Q]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)$
   if Q is Boolean

Proof: The proof is by structural induction on Q.

Base Case: For the base cases we have $\mathbf{Eval}\ [\![\mathbf{T}\ [\![V_i]\!] \texttt{ ! } Q']\!]\ \rho\ =\ \mathbf{Eval}\ [\![V_i]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)$ by **Lemma 4.1**. Also we have for non-Boolean OQL_{deB} constant or global name c,

$$
\begin{array}{rcl}
\mathbf{Eval}\ [\![\mathbf{T}\ [\![c]\!] \texttt{ ! } Q']\!]\ \rho & = & \mathbf{Eval}\ [\![\mathtt{K}_f\ (c) \texttt{ ! } Q']\!]\ \rho \\
& = & \mathbf{Eval}\ [\![c]\!]\ \rho \\
& = & c \\
& = & \mathbf{Eval}\ [\![c]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)
\end{array}
$$

Inductive Case: The inductive case must be proved by cases that vary according to the structure of Q.

Case: $Q \ \equiv \ \ll Q_0, \ldots, \ Q_m \gg$:

$$\begin{aligned}
&\textbf{Eval}\ [\![\textbf{T}\ [\![\ll Q_0, \ldots,\ Q_m \gg]\!]\ !\ Q']\!]\ \rho \\
&\quad = \ \textbf{Eval}\ [\![\langle \ldots \langle \textbf{T}\ [\![Q_0]\!],\ \textbf{T}\ [\![Q_1]\!]\rangle,\ \ldots,\ \textbf{T}\ [\![Q_m]\!]\rangle\ !\ Q']\!]\ \rho \\
&\quad = \ \textbf{Eval}\ [\![\ll \textbf{T}\ [\![Q_0]\!]\ !\ Q', \ldots,\ \textbf{T}\ [\![Q_m]\!]\ !\ Q' \gg]\!]\ \rho \\
&\quad = \ (\textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ !\ Q']\!]\ \rho, \ldots,\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_m]\!]\ !\ Q']\!]\ \rho) \\
&\quad = \ (\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho), \ldots,\ \textbf{Eval}\ [\![Q_m]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)) \ \ \text{(by induction)} \\
&\quad = \ \textbf{Eval}\ [\![\ll Q_0, \ldots,\ Q_m \gg]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}$$

Case: $Q \ \equiv \ Q_0\ .\ a_0$ (if $\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho) \ = \ (\delta,\ v)$)

$$\begin{aligned}
\textbf{Eval}\ [\![\textbf{T}\ [\![Q_0\ .\ a_0]\!]\ !\ Q']\!]\ \rho \ &= \ \textbf{Eval}\ [\![\pi_1 \ \circ\ \textbf{T}\ [\![Q_0]\!]\ !\ Q']\!]\ \rho \\
&= \ \textbf{Eval}\ [\![\pi_1\ !\ (\textbf{T}\ [\![Q_0]\!]\ !\ Q')]\!]\ \rho \\
&= \ \textbf{Eval}\ [\![(\textbf{T}\ [\![Q_0]\!]\ !\ Q')\ .\ a_0]\!]\ \rho \\
&= \ \delta, \ \ \text{if}\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ !\ Q']\!]\ \rho \ = \ (\delta,\ v) \\
&= \ \delta, \ \ \text{if}\ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho) \ = \ (\delta,\ v) \ \ \text{(by induction)} \\
&= \ \textbf{Eval}\ [\![Q_0\ .\ a_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}$$

Case: $Q \ \equiv \ Q_0\ .\ a_1$ (if $\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho) \ = \ (\delta,\ v)$)

$$\begin{aligned}
\textbf{Eval}\ [\![\textbf{T}\ [\![Q_0\ .\ a_1]\!]\ !\ Q']\!]\ \rho \ &= \ \textbf{Eval}\ [\![\pi_2 \ \circ\ \textbf{T}\ [\![Q_0]\!]\ !\ Q']\!]\ \rho \\
&= \ \textbf{Eval}\ [\![\pi_2\ !\ (\textbf{T}\ [\![Q_0]\!]\ !\ Q')]\!]\ \rho \\
&= \ \textbf{Eval}\ [\![(\textbf{T}\ [\![Q_0]\!]\ !\ Q')\ .\ a_1]\!]\ \rho \\
&= \ v, \ \ \text{if}\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ !\ Q']\!]\ \rho \ = \ (\delta,\ v) \\
&= \ v, \ \ \text{if}\ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho) \ = \ (\delta,\ v) \ \ \text{(by induction)} \\
&= \ \textbf{Eval}\ [\![Q_0\ .\ a_1]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}$$

Case: $Q \ \equiv \ Q_0 \rightarrow \texttt{att}$, if *att* is an attribute defined in the type of Q_0

$$\begin{aligned}
\textbf{Eval}\ [\![\textbf{T}\ [\![Q_0 \rightarrow\ \texttt{att}]\!]\ !\ Q']\!]\ \rho \ &= \ \textbf{Eval}\ [\![\texttt{att}\ \circ\ \textbf{T}\ [\![Q_0]\!]\ !\ Q']\!]\ \rho \\
&= \ \textbf{Eval}\ [\![\texttt{att}\ !\ (\textbf{T}\ [\![Q_0]\!]\ !\ Q')]\!]\ \rho \\
&= \ \textbf{Eval}\ [\![\textbf{T}\ [\![\texttt{att}]\!]\ !\ (\textbf{T}\ [\![Q_0]\!]\ !\ Q')]\!]\ \rho \\
&= \ \mathit{att}\ (\textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ !\ Q']\!]\ \rho) \ \ \text{(by induction)} \\
&= \ \mathit{att}\ (\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)) \ \ \text{(by induction)} \\
&= \ \textbf{Eval}\ [\![Q_0 \rightarrow\ \texttt{att}]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}$$

Case: $Q \ \equiv \ -\ Q_0$

$$\begin{aligned}
\textbf{Eval}\ [\![\textbf{T}\ [\![-\ Q_0]\!]\ !\ Q']\!]\ \rho \ &= \ \textbf{Eval}\ [\![(\textbf{inverse}\ \circ\ \textbf{T}\ [\![Q_0]\!])\ !\ Q']\!]\ \rho \\
&= \ \textbf{Eval}\ [\![\textbf{inverse}\ !\ (\textbf{T}\ [\![Q_0]\!]\ !\ Q')]\!]\ \rho
\end{aligned}$$

$$
\begin{aligned}
&= \textbf{Eval}\ \llbracket -\ (\mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q')\rrbracket\ \rho \\
&= -\ (\textbf{Eval}\ \llbracket \mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q' \rrbracket\ \rho) \\
&= -\ (\textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho)) \quad \text{(by induction)} \\
&= \textbf{Eval}\ \llbracket -\ Q_0 \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho)
\end{aligned}
$$

Case: $Q \equiv \texttt{abs}\ (Q_0)$

$$
\begin{aligned}
\textbf{Eval}\ \llbracket \mathbf{T}\ \llbracket \texttt{abs}\ (Q_0) \rrbracket\ \texttt{!}\ Q' \rrbracket\ \rho &= \textbf{Eval}\ \llbracket (\textbf{abs}\ \circ\ \mathbf{T}\ \llbracket Q_0 \rrbracket)\ \texttt{!}\ Q' \rrbracket\ \rho \\
&= \textbf{Eval}\ \llbracket \textbf{abs}\ \texttt{!}\ (\mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q')\rrbracket\ \rho \\
&= \textbf{Eval}\ \llbracket \texttt{abs}\ (\mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q')\rrbracket\ \rho \\
&= |\textbf{Eval}\ \llbracket \mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q' \rrbracket\ \rho| \\
&= |\textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho))| \quad \text{(by induction)} \\
&= \textbf{Eval}\ \llbracket \texttt{abs}\ (Q_0) \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho)
\end{aligned}
$$

Case: $Q \equiv Q_0\ +\ Q_1$ (similarly for $-$, $*$, $/$ and $\texttt{mod}$)

$$
\begin{aligned}
\textbf{Eval}\ \llbracket \mathbf{T}\ \llbracket Q_0\ +\ Q_1 \rrbracket\ \texttt{!}\ Q' \rrbracket\ \rho &= \textbf{Eval}\ \llbracket (\textbf{add}\ \circ\ \langle \mathbf{T}\ \llbracket Q_0 \rrbracket,\ \mathbf{T}\ \llbracket Q_1 \rrbracket \rangle)\ \texttt{!}\ Q' \rrbracket\ \rho \\
&= \textbf{Eval}\ \llbracket \textbf{add}\ \texttt{!}\ (\langle \mathbf{T}\ \llbracket Q_0 \rrbracket,\ \mathbf{T}\ \llbracket Q_1 \rrbracket \rangle\ \texttt{!}\ Q')\rrbracket\ \rho \\
&= \textbf{Eval}\ \llbracket \textbf{add}\ \texttt{!}\ \ll \mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q',\ \mathbf{T}\ \llbracket Q_1 \rrbracket\ \texttt{!}\ Q' \gg \rrbracket\ \rho \\
&= \textbf{Eval}\ \llbracket (\mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q')\ +\ (\mathbf{T}\ \llbracket Q_1 \rrbracket\ \texttt{!}\ Q')\rrbracket\ \rho \\
&= (\textbf{Eval}\ \llbracket \mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q' \rrbracket\ \rho) + (\textbf{Eval}\ \llbracket \mathbf{T}\ \llbracket Q_1 \rrbracket\ \texttt{!}\ Q' \rrbracket\ \rho) \\
&= (\textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho)) + (\textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho))\ \text{(by induction)} \\
&= \textbf{Eval}\ \llbracket Q_0\ +\ Q_1 \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho)
\end{aligned}
$$

Case: $Q \equiv Q_0\ \texttt{intersect}\ Q_1$ (similarly for $\texttt{union}$ and $\texttt{except}$)

$$
\begin{aligned}
&\textbf{Eval}\ \llbracket \mathbf{T}\ \llbracket Q_0\ \texttt{intersect}\ Q_1 \rrbracket\ \texttt{!}\ Q' \rrbracket\ \rho \\
&\quad = \textbf{Eval}\ \llbracket (\textbf{int}\ (\textbf{eq})\ \circ\ \langle \mathbf{T}\ \llbracket Q_0 \rrbracket,\ \mathbf{T}\ \llbracket Q_1 \rrbracket \rangle)\ \texttt{!}\ Q' \rrbracket\ \rho \\
&\quad = \textbf{Eval}\ \llbracket \textbf{int}\ (\textbf{eq})\ \texttt{!}\ (\langle \mathbf{T}\ \llbracket Q_0 \rrbracket,\ \mathbf{T}\ \llbracket Q_1 \rrbracket \rangle\ \texttt{!}\ Q')\rrbracket\ \rho \\
&\quad = \textbf{Eval}\ \llbracket \textbf{int}\ (\textbf{eq})\ \texttt{!}\ \ll \mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q',\ \mathbf{T}\ \llbracket Q_1 \rrbracket\ \texttt{!}\ Q' \gg \rrbracket\ \rho
\end{aligned}
$$

$$
= \textbf{Eval} \left\llbracket
\begin{array}{l}
\texttt{select distinct}\ V_0 \\
\texttt{from}\ \mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q' \\
\texttt{where exists}\ ((\mathbf{T}\ \llbracket Q_1 \rrbracket\ \texttt{!}\ Q') \uparrow \tbinom{0}{1})\ :\ \textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg \\
\qquad\qquad\qquad\qquad \texttt{intersect} \\
\texttt{select distinct}\ V_0 \\
\texttt{from}\ \mathbf{T}\ \llbracket Q_1 \rrbracket\ \texttt{!}\ Q' \\
\texttt{where exists}\ ((\mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q') \uparrow \tbinom{0}{1})\ :\ \textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg
\end{array}
\right\rrbracket \rho
$$

$$
= \begin{array}{c}
\textbf{Eval} \left\llbracket
\begin{array}{l}
\texttt{select distinct}\ V_0 \\
\texttt{from}\ \mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q' \\
\texttt{where exists}\ ((\mathbf{T}\ \llbracket Q_1 \rrbracket\ \texttt{!}\ Q') \uparrow \tbinom{0}{1})\ :\ \textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg
\end{array}
\right\rrbracket \rho \\
\cap \\
\textbf{Eval} \left\llbracket
\begin{array}{l}
\texttt{select distinct}\ V_0 \\
\texttt{from}\ \mathbf{T}\ \llbracket Q_1 \rrbracket\ \texttt{!}\ Q' \\
\texttt{where exists}\ ((\mathbf{T}\ \llbracket Q_0 \rrbracket\ \texttt{!}\ Q') \uparrow \tbinom{0}{1})\ :\ \textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg
\end{array}
\right\rrbracket \rho
\end{array}
$$

$$= \begin{array}{c} \left\{ \begin{array}{l} \textbf{Eval}\ [\![V_0]\!]\ (\rho,\ x_0) \mid x_0\ \in\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho, \\ \ \textbf{Eval}\ [\![\texttt{exists}\ ((\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q') \uparrow \binom{0}{1})\ \texttt{:}\ \textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0)\ =\ \mathit{true} \end{array} \right\} \\ \cap \\ \left\{ \begin{array}{l} \textbf{Eval}\ [\![V_0]\!]\ (\rho,\ x_0) \mid x_0\ \in\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho, \\ \ \textbf{Eval}\ [\![\texttt{exists}\ ((\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \binom{0}{1})\ \texttt{:}\ \textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0)\ =\ \mathit{true} \end{array} \right\} \end{array}$$

$$= \begin{array}{c} \left\{ \begin{array}{l} x_0 \mid x_0\ \in\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho, \\ \ \exists\ x\ \in\ \textbf{Eval}\ [\![(\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q') \uparrow \binom{0}{1}]\!]\ (\rho,\ x_0)\ (\textbf{Eval}\ [\![\textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0,\ x)\ =\ \mathit{true}) \end{array} \right\} \\ \cap \\ \left\{ \begin{array}{l} x_0 \mid x_0\ \in\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho, \\ \ \exists\ x\ \in\ \textbf{Eval}\ [\![(\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \binom{0}{1}]\!]\ (\rho,\ x_0)\ (\textbf{Eval}\ [\![\textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0,\ x)\ =\ \mathit{true}) \end{array} \right\} \end{array}$$

$$= \begin{array}{c} \left\{ \begin{array}{l} x_0 \mid x_0\ \in\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho, \\ \ \exists\ x\ \in\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval}\ [\![\textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0,\ x)\ =\ \mathit{true}) \end{array} \right\} \\ \cap \\ \left\{ \begin{array}{l} x_0 \mid x_0\ \in\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho, \\ \ \exists\ x\ \in\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval}\ [\![\textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0,\ x)\ =\ \mathit{true}) \end{array} \right\} \end{array}$$

(By Lemma 3.1)

$$= \begin{array}{c} \left\{ \begin{array}{l} x_0 \mid x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho), \\ \ \exists\ x\ \in\ \textbf{Eval}\ [\![Q_1]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)\ (\textbf{Eval}\ [\![\textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0,\ x)\ =\ \mathit{true}) \end{array} \right\} \\ \cap \\ \left\{ \begin{array}{l} x_0 \mid x_0\ \in\ \textbf{Eval}\ [\![Q_1]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho), \\ \ \exists\ x\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)\ (\textbf{Eval}\ [\![\textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0,\ x)\ =\ \mathit{true}) \end{array} \right\} \end{array}$$

(by induction)

$$= \begin{array}{c} \left\{ \begin{array}{l} x_0 \mid x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho), \\ \ \exists\ x\ \in\ \textbf{Eval}\ [\![Q_1 \uparrow \binom{1}{0}]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0)\ (\textbf{Eval}\ [\![\textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0,\ x)\ =\ \mathit{true}) \end{array} \right\} \\ \cap \\ \left\{ \begin{array}{l} x_0 \mid x_0\ \in\ \textbf{Eval}\ [\![Q_1]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho), \\ \ \exists\ x\ \in\ \textbf{Eval}\ [\![Q_0 \uparrow \binom{1}{0}]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0)\ (\textbf{Eval}\ [\![\textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg]\!]\ (\rho,\ x_0,\ x)\ =\ \mathit{true}) \end{array} \right\} \end{array}$$

(by induction)

$$= \textbf{Eval} \left[\!\!\left[\begin{array}{l} \texttt{select distinct}\ V_0 \\ \texttt{from}\ Q_0 \\ \texttt{where exists}\ (Q_1 \uparrow \binom{1}{0})\ \texttt{:}\ \textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg \\ \qquad\qquad\quad \texttt{intersect} \\ \texttt{select distinct}\ V_0 \\ \texttt{from}\ Q_1 \\ \texttt{where exists}\ (Q_0 \uparrow \binom{1}{0})\ \texttt{:}\ \textbf{eq}\ \texttt{?} \ll V_1,\ V_0 \gg \end{array} \right]\!\!\right] (\textbf{Eval}\ [\![Q']\!]\ \rho)$$

$$= \textbf{Eval}\ [\![Q_0\ \texttt{intersect}\ Q_1]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)$$

Case: $Q\ \equiv\ \texttt{set}\ (Q_0,\ \ldots,\ Q_n)$

$$\textbf{Eval}\ [\![\textbf{T}\ [\![\texttt{set}\ (Q_0,\ \ldots,\ Q_n)]\!]\ \texttt{!}\ Q']\!]\ \rho$$

$$= \textbf{Eval}\ [\![(\textbf{ins} \circ \langle \textbf{T}\ [\![Q_0]\!],\ \textbf{ins} \circ \ldots \circ\ \textbf{ins} \circ \langle \textbf{T}\ [\![Q_n]\!],\ \mathtt{K}_f\ (\oslash) \rangle \ldots \rangle)\ \texttt{!}\ Q']\!]\ \rho$$

$$
\begin{aligned}
&= \textbf{Eval}\ [\![\texttt{set}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')\ \texttt{union}\ (\ldots\ ((\texttt{set}\ (\textbf{T}\ [\![Q_n]\!]\ \texttt{!}\ Q')\ \texttt{union}\ \oslash)\ldots)]\!]\ \rho\\
&= \textbf{Eval}\ [\![\texttt{set}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho\ \cup\ (\ldots\ (\textbf{Eval}\ [\![\texttt{set}\ (\textbf{T}\ [\![Q_n]\!]\ \texttt{!}\ Q')]\!]\ \rho\ \cup\ \textbf{Eval}\ [\![\oslash]\!]\ \rho)\ldots)\\
&= \{\textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho\}\ \cup\ (\ldots\ (\{\textbf{Eval}\ [\![\textbf{T}\ [\![Q_n]\!]\ \texttt{!}\ Q']\!]\ \rho\}\ \cup\ \oslash)\ldots)\\
&= \{\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)\}\ \cup\ (\ldots\ (\{\textbf{Eval}\ [\![Q_n]\!](\textbf{Eval}\ [\![Q']\!]\ \rho)\}\ \cup\ \oslash)\ldots)\ \ \text{(by induction)}\\
&= \{\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho),\ \ldots,\ \textbf{Eval}\ [\![Q_n]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)\}\\
&= \textbf{Eval}\ [\![\texttt{set}\ (Q_0,\ \ldots,\ Q_n)]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q \equiv \texttt{element}\ (Q_0)$, if $\textbf{Eval}\ [\![Q_0]\!]\ \rho = \{x\}$ for some $x \in \mathcal{D}$

$$
\begin{aligned}
\textbf{Eval}\ [\![\textbf{T}\ [\![\texttt{element}\ (Q_0)]\!]\ \texttt{!}\ Q']\!]\ \rho &= \textbf{Eval}\ [\![(\textbf{choose} \circ \textbf{T}\ [\![Q_0]\!])\ \texttt{!}\ Q']\!]\ \rho\\
&= \textbf{Eval}\ [\![\textbf{choose}\ \texttt{!}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho\\
&= \textbf{Eval}\ [\![\texttt{element}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho\\
&= x,\ \text{if}\ \textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho\ =\ \{x\}\\
&= x,\ \text{if}\ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)\ =\ \{x\}\ \text{(by induction)}\\
&= \textbf{Eval}\ [\![\texttt{element}\ (Q_0)]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q \equiv \texttt{flatten}\ (Q_0)$

$$
\begin{aligned}
\textbf{Eval}\ [\![\textbf{T}\ [\![\texttt{flatten}\ (Q_0)]\!]\ \texttt{!}\ Q']\!]\ \rho &= \textbf{Eval}\ [\![(\textbf{flat} \circ \textbf{T}\ [\![Q_0]\!])\ \texttt{!}\ Q']\!]\ \rho\\
&= \textbf{Eval}\ [\![\textbf{flat}\ \texttt{!}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho\\
&= \textbf{Eval}\ [\![\texttt{flatten}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho\\
&= \{y \mid \exists\, x\ \in\ (\textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho)\ (y\ \in\ x)\}\\
&= \{y \mid \exists\, x\ \in\ (\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho))\ (y\ \in\ x)\}\ \text{(by induction)}\\
&= \textbf{Eval}\ [\![\texttt{flatten}\ (Q_0)]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q \equiv \texttt{count}\ (Q_0)$ (similarly for `sum` and `avg`.)

$$
\begin{aligned}
\textbf{Eval}\ [\![\textbf{T}\ [\![\texttt{count}\ (Q_0)]\!]\ \texttt{!}\ Q']\!]\ \rho &= \textbf{Eval}\ [\![\overline{\texttt{count}} \circ \textbf{T}\ [\![Q_0]\!])\ \texttt{!}\ Q']\!]\ \rho\\
&= \textbf{Eval}\ [\![\overline{\texttt{count}}\ \texttt{!}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho\\
&= \textbf{Eval}\ [\![\texttt{count}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho\\
&= |\textbf{Eval}\ [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho|\\
&= |\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)|\ \ \text{(by induction)}\\
&= \textbf{Eval}\ [\![\texttt{count}\ (Q_0)]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q \equiv \texttt{min}\ (Q_0)$ (similarly for `max`.)

$$
\begin{aligned}
&\textbf{Eval}\ [\![\textbf{T}\ [\![\texttt{min}\ (Q_0)]\!]\ \texttt{!}\ Q']\!]\ \rho\\
&\quad= \textbf{Eval}\ [\![\textbf{bound}\ (\textbf{leq}) \circ \textbf{T}\ [\![Q_0]\!])\ \texttt{!}\ Q']\!]\ \rho\\
&\quad= \textbf{Eval}\ [\![\textbf{bound}\ (\textbf{leq})\ \texttt{!}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho
\end{aligned}
$$

$$= \textbf{Eval} \left[\!\left[\texttt{element} \left(\begin{array}{l} \texttt{select distinct } V_0 \\ \texttt{from } \mathbf{T} [\![Q_0]\!] \texttt{ ! } Q' \\ \texttt{where (for\_all } ((\mathbf{T} [\![Q_0]\!] \texttt{ ! } Q') \uparrow \binom{1}{0}) \texttt{ : } \mathbf{leq} \texttt{ ? } \ll V_1,\ V_0 \gg) \end{array} \right) \right]\!\right] \rho$$

$$= x, \text{ if } \textbf{Eval} \left[\!\left[\begin{array}{l} \texttt{select distinct } V_0 \\ \texttt{from } \mathbf{T} [\![Q_0]\!] \texttt{ ! } Q' \\ \texttt{where (for\_all } ((\mathbf{T} [\![Q_0]\!] \texttt{ ! } Q') \uparrow \binom{1}{0}) \texttt{ : } \mathbf{leq} \texttt{ ? } \ll V_1,\ V_0 \gg) \end{array} \right]\!\right] \rho \ = \ \{x\}$$

$$= x, \text{ if } \left\{ \begin{array}{l} \textbf{Eval } [\![V_0]\!] (\rho,\ y) \mid y \ \in \ \textbf{Eval } [\![\mathbf{T} [\![Q_0]\!] \texttt{ ! } Q']\!] \rho, \\ \textbf{Eval } [\![\texttt{for\_all } ((\mathbf{T} [\![Q_0]\!] \texttt{ ! } Q') \uparrow \binom{1}{0}) \texttt{ : } \mathbf{leq} \texttt{ ? } \ll V_1,\ V_0 \gg]\!] (\rho,\ y) \ = \ \mathit{true} \end{array} \right\} \ = \ \{x\}$$

$$= x, \text{ if } \left\{ \begin{array}{l} y \mid y \ \in \ \textbf{Eval } [\![\mathbf{T} [\![Q_0]\!] \texttt{ ! } Q']\!] \rho, \\ \forall\, z \ \in \ \textbf{Eval } [\![(\mathbf{T} [\![Q_0]\!] \texttt{ ! } Q') \uparrow \binom{1}{0}]\!] (\rho,\ y) \ (\textbf{Eval } [\![\mathbf{leq} \texttt{ ? } \ll V_1,\ V_0 \gg]\!] (\rho,\ y,\ z) \ = \ \mathit{true}) \end{array} \right\}$$
$$= \ \{x\}$$

$$= x, \text{ if } \left\{ \begin{array}{l} y \mid y \ \in \ \textbf{Eval } [\![\mathbf{T} [\![Q_0]\!] \texttt{ ! } Q']\!] \rho, \\ \forall\, z \ \in \ \textbf{Eval } [\![\mathbf{T} [\![Q_0]\!] \texttt{ ! } Q']\!] \rho \ (\textbf{Eval } [\![\mathbf{leq} \texttt{ ? } \ll V_1,\ V_0 \gg]\!] (\rho,\ y,\ z) \ = \ \mathit{true}) \end{array} \right\} \ = \ \{x\}$$

(By Lemma 3.1)

$$= x, \text{ if } \left\{ \begin{array}{l} y \mid y \ \in \ \textbf{Eval } [\![Q_0]\!] (\textbf{Eval } [\![Q']\!] \rho), \\ \forall\, z \ \in \ \textbf{Eval } [\![Q_0]\!] (\textbf{Eval } [\![Q']\!] \rho) \ (\textbf{Eval } [\![V_1 \ \leq \ V_0]\!] (\rho,\ y,\ z) \ = \ \mathit{true}) \end{array} \right\} \ = \ \{x\}$$

$$\begin{aligned}
&= x, \text{ if } \{y \mid y \ \in \ \textbf{Eval } [\![Q_0]\!] (\textbf{Eval } [\![Q']\!] \rho),\ \forall\, z \ \in \ \textbf{Eval } [\![Q_0]\!] (\textbf{Eval } [\![Q']\!] \rho) \ (y \ \leq \ z)\} \ = \ \{x\} \\
&= x, \text{ if } x \ \in \ \textbf{Eval } [\![Q_0]\!] (\textbf{Eval } [\![Q']\!] \rho),\ \forall\, z \ \in \ \textbf{Eval } [\![Q_0]\!] (\textbf{Eval } [\![Q']\!] \rho) \ (x \ \leq \ z) \\
&= \textbf{Eval } [\![\texttt{min } (Q_0)]\!] (\textbf{Eval } [\![Q']\!] \rho)
\end{aligned}$$

Case: $Q \ \equiv \ \texttt{select distinct } Q_s \texttt{ from } Q_0,\ \ldots,\ Q_m \texttt{ where } Q_w$

$$\textbf{Eval} \left[\!\left[\mathbf{T} \left[\!\left[\begin{array}{l} \texttt{select distinct } Q_s \\ \texttt{from } Q_0,\ \ldots,\ Q_m \\ \texttt{where } Q_w \end{array} \right]\!\right] \texttt{ ! } Q' \right]\!\right] \rho$$

$$= \quad \textbf{Eval} \left[\!\left[\left(\begin{array}{l} \mathbf{iterate} \ (\mathbf{T} [\![Q_w]\!],\ \mathbf{T} [\![Q_s]\!]) \circ \\ \ \mathbf{flat} \circ \mathbf{iterate} \ (\texttt{K}_p \ (\texttt{T}),\ \mathbf{iter} \ (\texttt{K}_p \ (\texttt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathbf{T} [\![Q_m]\!] \rangle) \circ \\ \ \ \ldots \circ \\ \ \ \mathbf{flat} \circ \mathbf{iterate} \ (\texttt{K}_p \ (\texttt{T}),\ \mathbf{iter} \ (\texttt{K}_p \ (\texttt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathbf{T} [\![Q_0]\!] \rangle) \circ \\ \ \ \mathbf{set} \end{array} \right) \texttt{ ! } Q' \right]\!\right] \rho$$

$$= \quad \textbf{Eval} \left[\!\left[\begin{array}{l} \mathbf{iterate} \ (\mathbf{T} [\![Q_w]\!],\ \mathbf{T} [\![Q_s]\!]) \texttt{ ! } \\ \quad \left(\begin{array}{l} \mathbf{flat} \circ \mathbf{iterate} \ (\texttt{K}_p \ (\texttt{T}),\ \mathbf{iter} \ (\texttt{K}_p \ (\texttt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathbf{T} [\![Q_m]\!] \rangle) \circ \\ \ \ldots \circ \\ \ \mathbf{flat} \circ \mathbf{iterate} \ (\texttt{K}_p \ (\texttt{T}),\ \mathbf{iter} \ (\texttt{K}_p \ (\texttt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathbf{T} [\![Q_0]\!] \rangle) \circ \\ \ \mathbf{set} \end{array} \right) \texttt{ ! } Q' \end{array} \right]\!\right] \rho$$

$$= \mathbf{Eval}\left[\!\!\left[\begin{array}{l}\texttt{iterate}\,(\mathbf{T}\,[\![Q_w]\!],\ \mathbf{T}\,[\![Q_s]\!])\ \texttt{!}\\ \left(\begin{array}{l}\texttt{select distinct} \ll V_{m+1},\ldots,\ V_0\gg\\ \texttt{from}\ \left(\begin{array}{l}\texttt{set}\,(Q'),\\ \mathbf{T}\,[\![Q_0]\!]\ \texttt{!}\ V_0,\\ ((\mathbf{T}\,[\![Q_1]\!]\ \texttt{!}\ V_0)\uparrow\binom{2}{1})\downarrow\binom{\ll V_1,\ldots,\ V_0\gg}{0},\ldots,\\ ((\mathbf{T}\,[\![Q_m]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+1}{1})\downarrow\binom{\ll V_m,\ldots,\ V_0\gg}{0}\end{array}\right)\end{array}\right)\end{array}\right]\!\!\right]\rho \quad \text{(by Lemma 4.2)}$$

$$= \mathbf{Eval}\left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ \mathbf{T}\,[\![Q_s]\!]\ \texttt{!}\ V_0\\ \texttt{from}\ \left(\begin{array}{l}\texttt{select distinct} \ll V_{m+1},\ldots,\ V_0\gg\\ \texttt{from}\ \left(\begin{array}{l}\texttt{set}\,(Q'),\\ \mathbf{T}\,[\![Q_0]\!]\ \texttt{!}\ V_0,\\ ((\mathbf{T}\,[\![Q_1]\!]\ \texttt{!}\ V_0)\uparrow\binom{2}{1})\downarrow\binom{\ll V_1,\ldots,\ V_0\gg}{0},\ldots,\\ ((\mathbf{T}\,[\![Q_m]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+1}{1})\downarrow\binom{\ll V_m,\ldots,\ V_0\gg}{0}\end{array}\right)\end{array}\right)\\ \texttt{where}\ \mathbf{T}\,[\![Q_w]\!]\ \texttt{?}\ V_0\end{array}\right]\!\!\right]\rho$$

$$= \mathbf{Eval}\left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ ((\mathbf{T}\,[\![Q_s]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+2}{1})\downarrow\binom{\ll V_{m+1},\ldots,\ V_0\gg}{0}\\ \texttt{from}\ \left(\begin{array}{l}\texttt{set}\,(Q'),\\ \mathbf{T}\,[\![Q_0]\!]\ \texttt{!}\ V_0,\\ ((\mathbf{T}\,[\![Q_1]\!]\ \texttt{!}\ V_0)\uparrow\binom{2}{1})\downarrow\binom{\ll V_1,\ldots,\ V_0\gg}{0},\ldots,\\ ((\mathbf{T}\,[\![Q_m]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+1}{1})\downarrow\binom{\ll V_m,\ldots,\ V_0\gg}{0}\end{array}\right)\\ \texttt{where}\ ((\mathbf{T}\,[\![Q_w]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+2}{1})\downarrow\binom{\ll V_{m+1},\ldots,\ V_0\gg}{0}\end{array}\right]\!\!\right]\rho$$

(By Lemma 3.7)

$$= \left\{\begin{array}{l}\mathbf{Eval}\ \left[\!\!\left[((\mathbf{T}\,[\![Q_s]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+2}{1})\downarrow\binom{\ll V_{m+1},\ldots,\ V_0\gg}{0}\right]\!\!\right]\ (\rho,\ x,\ x_0,\ldots,\ x_m)\ |\\ x\ \in\ \mathbf{Eval}\ [\![\texttt{set}\,(Q')]\!]\ \rho,\\ x_0\ \in\ \mathbf{Eval}\ [\![\mathbf{T}\,[\![Q_0]\!]\ \texttt{!}\ V_0]\!]\ (\rho,\ x),\\ x_1\ \in\ \mathbf{Eval}\ [\![((\mathbf{T}\,[\![Q_1]\!]\ \texttt{!}\ V_0)\uparrow\binom{2}{1})\downarrow\binom{\ll V_1,\ldots,\ V_0\gg}{0}]\!]\ (\rho,\ x,\ x_0),\ldots,\\ x_m\ \in\ \mathbf{Eval}\ [\![((\mathbf{T}\,[\![Q_m]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+1}{1})\downarrow\binom{\ll V_m,\ldots,\ V_0\gg}{0}]\!]\ (\rho,\ x,\ x_0,\ldots,\ x_{m-1}),\\ \mathbf{Eval}\ \left[\!\!\left[((\mathbf{T}\,[\![Q_w]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+2}{1})\downarrow\binom{\ll V_{m+1},\ldots,\ V_0\gg}{0}\right]\!\!\right]\ (\rho,\ x,\ x_0,\ldots,\ x_m)\ =\ \mathit{true}\end{array}\right\}$$

$$= \left\{\begin{array}{l}\mathbf{Eval}\ [\![(\mathbf{T}\,[\![Q_s]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+2}{1}]\!]\\ \quad(\rho,\ x,\ x_0,\ldots,\ x_m,\ \mathbf{Eval}\ [\![\ll V_{m+1},\ldots,\ V_0\gg]\!]\ (\rho,\ x,\ x_0,\ldots,\ x_m))\ |\\ x\ \in\ \mathbf{Eval}\ [\![\texttt{set}\,(Q')]\!]\ \rho,\\ x_0\ \in\ \mathbf{Eval}\ [\![\mathbf{T}\,[\![Q_0]\!]\ \texttt{!}\ V_0]\!]\ (\rho,\ x),\\ x_1\ \in\ \mathbf{Eval}\ [\![(\mathbf{T}\,[\![Q_1]\!]\ \texttt{!}\ V_0)\uparrow\binom{2}{1}]\!]\ (\rho,\ x,\ x_0,\ \mathbf{Eval}\ [\![\ll V_1,\ldots,\ V_0\gg]\!]\ (\rho,\ x,\ x_0)),\ldots,\\ x_m\ \in\ \mathbf{Eval}\ [\![(\mathbf{T}\,[\![Q_m]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+1}{1}]\!]\\ \quad(\rho,\ x,\ x_0,\ldots,\ x_{m-1},\ \mathbf{Eval}\ [\![\ll V_m,\ldots,\ V_0\gg]\!]\ (\rho,\ x,\ x_0,\ldots,\ x_{m-1})),\\ \mathbf{Eval}\ [\![(\mathbf{T}\,[\![Q_w]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+2}{1}]\!]\\ \quad(\rho,\ x,\ x_0,\ldots,\ x_m,\ \mathbf{Eval}\ [\![\ll V_{m+1},\ldots,\ V_0\gg]\!]\ (\rho,\ x,\ x_0,\ldots,\ x_m))\ =\ \mathit{true}\end{array}\right\}$$

(By Lemma 3.2)

$$= \left\{\begin{array}{l}\mathbf{Eval}\ [\![(\mathbf{T}\,[\![Q_s]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+2}{1}]\!]\ (\rho,\ x,\ x_0,\ldots,\ x_m,\ (x,\ x_0,\ldots,\ x_m))\ |\\ x\ \in\ \mathbf{Eval}\ [\![\texttt{set}\,(Q')]\!]\ \rho,\\ x_0\ \in\ \mathbf{Eval}\ [\![\mathbf{T}\,[\![Q_0]\!]\ \texttt{!}\ V_0]\!]\ (\rho,\ x),\\ x_1\ \in\ \mathbf{Eval}\ [\![(\mathbf{T}\,[\![Q_1]\!]\ \texttt{!}\ V_0)\uparrow\binom{2}{1}]\!]\ (\rho,\ x,\ x_0,\ (x,\ x_0)),\ldots,\\ x_m\ \in\ \mathbf{Eval}\ [\![(\mathbf{T}\,[\![Q_m]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+1}{1}]\!]\ (\rho,\ x,\ x_0,\ldots,\ x_{m-1},\ (x,\ x_0,\ldots,\ x_{m-1})),\\ \mathbf{Eval}\ [\![(\mathbf{T}\,[\![Q_w]\!]\ \texttt{!}\ V_0)\uparrow\binom{m+2}{1}]\!]\ (\rho,\ x,\ x_0,\ldots,\ x_m,\ (x,\ x_0,\ldots,\ x_m))\ =\ \mathit{true}\end{array}\right\}$$

$$= \left\{ \begin{array}{l} \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_s]\!]\ !\ V_0]\!]\ (\rho,\ (x,\ x_0, \ldots,\ x_m))\ | \\ \ x\ \in\ \mathbf{Eval}\ \ [\![\mathtt{set}\ (Q')]\!]\ \rho, \\ \ x_0\ \in\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_0]\!]\ !\ V_0]\!]\ (\rho,\ x), \\ \ x_1\ \in\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_1]\!]\ !\ V_0]\!]\ (\rho,\ (x,\ x_0)), \ldots, \\ \ x_m\ \in\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_m]\!]\ !\ V_0]\!]\ (\rho,\ (x,\ x_0, \ldots,\ x_{m-1})), \\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_w]\!]\ !\ V_0]\!]\ (\rho,\ (x,\ x_0, \ldots,\ x_m))\ =\ \mathit{true} \end{array} \right\}$$

(By Lemma 3.1)

$$= \left\{ \begin{array}{l} \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_s]\!]\ !\ V_0]\!]\ (\rho,\ (x,\ x_0, \ldots,\ x_m))\ | \\ \ x\ \in\ \{\mathbf{Eval}\ \ [\![Q']\!]\ \rho\}, \\ \ x_0\ \in\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_0]\!]\ !\ V_0]\!]\ (\rho,\ x), \\ \ x_1\ \in\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_1]\!]\ !\ V_0]\!]\ (\rho,\ (x,\ x_0)), \ldots, \\ \ x_m\ \in\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_m]\!]\ !\ V_0]\!]\ (\rho,\ (x,\ x_0, \ldots,\ x_{m-1})), \\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_w]\!]\ !\ V_0]\!]\ (\rho,\ (x,\ x_0, \ldots,\ x_m))\ =\ \mathit{true} \end{array} \right\}$$

$$= \left\{ \begin{array}{l} \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_s]\!]\ !\ V_0]\!]\ (\rho,\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m))\ | \\ \ x_0\ \in\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_0]\!]\ !\ V_0]\!]\ (\rho,\ \mathbf{Eval}\ \ [\![Q']\!]\ \rho), \\ \ x_1\ \in\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_1]\!]\ !\ V_0]\!]\ (\rho,\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0)), \ldots, \\ \ x_m\ \in\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_m]\!]\ !\ V_0]\!]\ (\rho,\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_{m-1})), \\ \mathbf{Eval}\ \ [\![\mathbf{T}\ [\![Q_w]\!]\ !\ V_0]\!]\ (\rho,\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m))\ =\ \mathit{true} \end{array} \right\}$$

$$= \left\{ \begin{array}{l} \mathbf{Eval}\ \ [\![Q_s]\!]\ (\mathbf{Eval}\ \ [\![V_0]\!]\ (\rho,\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)))\ | \\ \ x_0\ \in\ \mathbf{Eval}\ \ [\![Q_0]\!]\ (\mathbf{Eval}\ \ [\![V_0]\!]\ (\rho,\ \mathbf{Eval}\ \ [\![Q']\!]\ \rho)), \\ \ x_1\ \in\ \mathbf{Eval}\ \ [\![Q_1]\!]\ (\mathbf{Eval}\ \ [\![V_0]\!]\ (\rho,\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0))), \ldots, \\ \ x_m\ \in\ \mathbf{Eval}\ \ [\![Q_m]\!]\ (\mathbf{Eval}\ \ [\![V_0]\!]\ (\rho,\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_{m-1}))), \\ \mathbf{Eval}\ \ [\![Q_w]\!]\ (\mathbf{Eval}\ \ [\![V_0]\!]\ (\rho,\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)))\ =\ \mathit{true} \end{array} \right\}$$

(by induction)

$$= \left\{ \begin{array}{l} \mathbf{Eval}\ \ [\![Q_s]\!]\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)\ | \\ \ x_0\ \in\ \mathbf{Eval}\ \ [\![Q_0]\!]\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho), \\ \ x_1\ \in\ \mathbf{Eval}\ \ [\![Q_1]\!]\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0), \ldots, \\ \ x_m\ \in\ \mathbf{Eval}\ \ [\![Q_m]\!]\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_{m-1}), \\ \mathbf{Eval}\ \ [\![Q_w]\!]\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)\ =\ \mathit{true} \end{array} \right\}$$

$$= \quad \mathbf{Eval}\ \left[\!\!\left[\begin{array}{l} \mathtt{select\ distinct}\ Q_s \\ \mathtt{from}\ Q_0,\ \ldots,\ Q_m \\ \mathtt{where}\ Q_w \end{array} \right]\!\!\right]\ (\mathbf{Eval}\ \ [\![Q']\!]\ \rho)$$

Case: $Q\ \equiv\ \mathtt{select\ distinct}\ Q_s\ \mathtt{from}\ Q_0,\ \ldots,\ Q_m\ \mathtt{where}\ Q_w\ \mathtt{groupby}\ Q_{m+1}, \ldots,\ Q_p$

$$\mathbf{Eval}\ \left[\!\!\left[\mathbf{T}\ \left[\!\!\left[\begin{array}{l} \mathtt{select\ distinct}\ Q_s \\ \mathtt{from}\ Q_0,\ \ldots,\ Q_m \\ \mathtt{where}\ Q_w \\ \mathtt{groupby}\ Q_{m+1}, \ldots,\ Q_p \end{array} \right]\!\!\right]\ !\ Q' \right]\!\!\right]\ \rho$$

$$= \mathbf{Eval} \left\llbracket \left(\begin{array}{l} \mathbf{grp}\ (\mathbf{eq}\ \oplus\ (\pi_1\ \times\ \pi_1),\ \pi_1\ \times\ \mathbf{id},\ \pi_2) \circ \\ \mathbf{T} \left\llbracket \begin{array}{l} \texttt{select distinct}\ \ll \diamond,\ f_{m+1},\ \ldots,\ f_p,\ Q_s \gg \\ \texttt{from}\ Q_0, \ldots, Q_m \\ \texttt{where}\ Q_w \end{array} \right\rrbracket \end{array} \right)\ \texttt{!}\ Q' \right\rrbracket \rho$$

$$\text{(where } f_i\ =\ \texttt{if}\ ((Q_i \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q_s}{0})\ \texttt{then 1 else 0)}$$

$$= \mathbf{Eval}\ \llbracket \mathbf{grp}\ (\mathbf{eq}\ \oplus\ (\pi_1\ \times\ \pi_1),\ \pi_1\ \times\ \mathbf{id},\ \pi_2)\ \texttt{!}\ A \rrbracket\ \rho$$

$$\text{(where } A\ =\ \mathbf{T} \left\llbracket \begin{array}{l} \texttt{select distinct}\ \ll \diamond,\ f_{m+1},\ \ldots,\ f_p,\ Q_s \gg \\ \texttt{from}\ Q_0, \ldots, Q_m \\ \texttt{where}\ Q_w \end{array} \right\rrbracket\ \texttt{!}\ Q',$$

$$\text{and } f_i\ =\ \texttt{if}\ ((Q_i \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q_s}{0})\ \texttt{then 1 else 0)}$$

$$= \mathbf{Eval} \left\llbracket \begin{array}{l} \texttt{select distinct} \\ \quad (\pi_1\ \times\ \mathbf{id})\ \texttt{!}\ \ll V_0,\ \left(\begin{array}{l} \texttt{select distinct}\ \pi_2\ \texttt{!}\ V_0 \\ \texttt{from}\ A \uparrow \tbinom{1}{0} \\ \texttt{where}\ (\mathbf{eq}\ \oplus\ (\pi_1\ \times\ \pi_1))\ \texttt{?}\ \ll V_1,\ V_0 \gg \end{array} \right) \gg \\ \texttt{from}\ A \end{array} \right\rrbracket \rho$$

$$\text{(where } A\ =\ \mathbf{T} \left\llbracket \begin{array}{l} \texttt{select distinct}\ \ll \diamond\ f_{m+1},\ \ldots,\ f_p,\ Q_s \gg \\ \texttt{from}\ Q_0, \ldots, Q_m \\ \texttt{where}\ Q_w \end{array} \right\rrbracket\ \texttt{!}\ Q',$$

$$\text{and } f_i\ =\ \texttt{if}\ ((Q_i \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q_s}{0})\ \texttt{then 1 else 0)}$$

$$= \mathbf{Eval} \left\llbracket \begin{array}{l} \texttt{select distinct}\ \ll V_0\ .\ a_0,\ \left(\begin{array}{l} \texttt{select distinct}\ V_0\ .\ a_1 \\ \texttt{from}\ A \uparrow \tbinom{1}{0} \\ \texttt{where}\ V_1\ .\ a_0\ \ ==\ \ V_0\ .\ a_0 \end{array} \right) \gg \\ \texttt{from}\ A \end{array} \right\rrbracket \rho$$

$$\text{(where } A\ =\ \mathbf{T} \left\llbracket \begin{array}{l} \texttt{select distinct}\ \ll \diamond,\ f_{m+1},\ \ldots,\ f_p,\ Q_s \gg \\ \texttt{from}\ Q_0, \ldots, Q_m \\ \texttt{where}\ Q_w \end{array} \right\rrbracket\ \texttt{!}\ Q',$$

$$\text{and } f_i\ =\ \texttt{if}\ ((Q_i \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q_s}{0})\ \texttt{then 1 else 0)}$$

$$= \left\{ \begin{array}{l} (\mathbf{Eval}\ \llbracket V_0\ .\ a_0 \rrbracket\ (\rho,\ x),\ f\ (x))\ |\ x\ \in\ \mathbf{Eval}\ \llbracket A \rrbracket\ \rho, \\ \quad f\ (x)\ =\ \mathbf{Eval} \left\llbracket \begin{array}{l} \texttt{select distinct}\ V_0\ .\ a_1 \\ \texttt{from}\ A \uparrow \tbinom{1}{0} \\ \texttt{where}\ V_1\ .\ a_0\ \ ==\ \ V_0\ .\ a_0 \end{array} \right\rrbracket (\rho,\ x) \end{array} \right\}$$

$$\text{(where } A\ =\ \mathbf{T} \left\llbracket \begin{array}{l} \texttt{select distinct}\ \ll \diamond,\ f_{m+1},\ \ldots,\ f_p,\ Q_s \gg \\ \texttt{from}\ Q_0, \ldots, Q_m \\ \texttt{where}\ Q_w \end{array} \right\rrbracket\ \texttt{!}\ Q',$$

$$\text{and } f_i\ =\ \texttt{if}\ ((Q_i \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q_s}{0})\ \texttt{then 1 else 0)}$$

$$= \left\{ \begin{array}{l} (\mathbf{Eval}\ \llbracket V_0\ .\ a_0 \rrbracket\ (\rho,\ x),\ f\ (x))\ |\ x\ \in\ \mathbf{Eval}\ \llbracket A \rrbracket\ \rho, \\ \quad f\ (x)\ =\ \{\mathbf{Eval}\ \llbracket V_0\ .\ a_1 \rrbracket\ (\rho,\ x,\ y)\ |\ y\ \in\ \mathbf{Eval}\ \llbracket A \uparrow \tbinom{1}{0} \rrbracket\ (\rho,\ x),\ \mathbf{Eval}\ \llbracket V_1\ .\ a_0\ \ ==\ \ V_0\ .\ a_0 \rrbracket\ (\rho,\ x,\ y)\} \end{array} \right\}$$

$$\text{(where } \mathbf{Eval}\ \llbracket A \rrbracket\ \rho\ =\ \mathbf{Eval} \left\llbracket \mathbf{T} \left\llbracket \begin{array}{l} \texttt{select distinct}\ \ll \diamond,\ f_{m+1},\ \ldots,\ f_p,\ Q_s \gg \\ \texttt{from}\ Q_0, \ldots, Q_m \\ \texttt{where}\ Q_w \end{array} \right\rrbracket\ \texttt{!}\ Q' \right\rrbracket \rho,$$

$$\text{and } f_i\ =\ \texttt{if}\ ((Q_i \uparrow \tbinom{m+1}{1}) \downarrow \tbinom{Q_s}{0})\ \texttt{then 1 else 0)}$$

$$= \left\{ \begin{array}{l} (\textbf{Eval}\ [\![V_0\ .\ a_0]\!]\ (\rho,\ x),\ f\ (x))\ |\ x\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho, \\ \quad f\ (x)\ =\ \{\textbf{Eval}\ [\![V_0\ .\ a_1]\!]\ (\rho,\ x,\ y)\ |\ y\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho,\ \textbf{Eval}\ [\![V_1\ .\ a_0\ ==\ V_0\ .\ a_0]\!]\ (\rho,\ x,\ y)\} \end{array} \right\}$$

$$\text{(where } \textbf{Eval}\ [\![A]\!]\ \rho\ =\ \textbf{Eval}\ \left[\!\!\left[\textbf{T} \left[\!\!\left[\begin{array}{l} \texttt{select distinct} \ll \diamond,\ f_{m+1},\ \ldots,\ f_p,\ Q_s \gg \\ \texttt{from}\ Q_0,\ldots,Q_m \\ \texttt{where}\ Q_w \end{array} \right]\!\!\right]\ !\ Q' \right]\!\!\right]\ \rho,$$

and $f_i\ =\ \texttt{if}\ ((Q_i \uparrow \binom{m\,+\,1}{1}) \downarrow \binom{Q_s}{0})\ \texttt{then 1 else 0})$

(By Lemma 3.1)

$$= \left\{ \begin{array}{l} (\textbf{Eval}\ [\![V_0\ .\ a_0]\!]\ (\rho,\ x),\ f\ (x))\ |\ x\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho, \\ \quad f\ (x)\ =\ \{\textbf{Eval}\ [\![V_0\ .\ a_1]\!]\ (\rho,\ x,\ y)\ |\ y\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho,\ \textbf{Eval}\ [\![V_1\ .\ a_0\ ==\ V_0\ .\ a_0]\!]\ (\rho,\ x,\ y)\} \end{array} \right\}$$

$$\text{(where } \textbf{Eval}\ [\![A]\!]\ \rho\ =\ \textbf{Eval}\ \left[\!\!\left[\begin{array}{l} \texttt{select distinct} \ll \diamond,\ f_{m+1},\ \ldots,\ f_p,\ Q_s \gg \\ \texttt{from}\ Q_0,\ldots,Q_m \\ \texttt{where}\ Q_w \end{array} \right]\!\!\right]\ (\textbf{Eval}\ [\![Q']\!]\ \rho),$$

and $f_i\ =\ \texttt{if}\ ((Q_i \uparrow \binom{m\,+\,1}{1}) \downarrow \binom{Q_s}{0})\ \texttt{then 1 else 0})$

(by induction)

$$= \left\{ \begin{array}{l} (\textbf{Eval}\ [\![V_0\ .\ a_0]\!]\ (\rho,\ x),\ f\ (x))\ |\ x\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho, \\ \quad f\ (x)\ =\ \{\textbf{Eval}\ [\![V_0\ .\ a_1]\!]\ (\rho,\ x,\ y)\ |\ y\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho,\ \textbf{Eval}\ [\![V_1\ .\ a_0\ ==\ V_0\ .\ a_0]\!]\ (\rho,\ x,\ y)\} \end{array} \right\}$$

(where $\textbf{Eval}\ [\![A]\!]\ \rho\ =$

$$\left\{ \begin{array}{l} (\epsilon,\ e_{m+1},\ldots,\ e_p,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m))\ | \\ \quad e_{m+1}\ =\ \textbf{Eval}\ [\![\texttt{if}\ ((Q_{m+1} \uparrow \binom{m\,+\,1}{1}) \downarrow \binom{Q_s}{0})\ \texttt{then 1 else 0}]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m),\ldots, \\ \quad e_p\ =\ \textbf{Eval}\ [\![\texttt{if}\ ((Q_p \uparrow \binom{m\,+\,1}{1}) \downarrow \binom{Q_s}{0})\ \texttt{then 1 else 0}]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m), \\ \quad x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho),\ldots, \\ \quad x_m\ \in\ \textbf{Eval}\ [\![Q_m]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_{m-1}), \\ \quad \textbf{Eval}\ [\![Q_w]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m)\ =\ \mathit{true} \end{array} \right\},$$

$$= \left\{ \begin{array}{l} (\textbf{Eval}\ [\![V_0\ .\ a_0]\!]\ (\rho,\ x),\ f\ (x))\ |\ x\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho, \\ \quad f\ (x)\ =\ \{\textbf{Eval}\ [\![V_0\ .\ a_1]\!]\ (\rho,\ x,\ y)\ |\ y\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho,\ \textbf{Eval}\ [\![V_1\ .\ a_0\ ==\ V_0\ .\ a_0]\!]\ (\rho,\ x,\ y)\} \end{array} \right\}$$

(where $\textbf{Eval}\ [\![A]\!]\ \rho\ =$

$$\left\{ \begin{array}{l} (\epsilon,\ e_{m+1},\ldots,\ e_p,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m))\ | \\ \quad e_{m+1}\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \textbf{Eval}\ [\![(Q_{m+1} \uparrow \binom{m\,+\,1}{1}) \downarrow \binom{Q_s}{0}]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m)\ =\ \mathit{true}, \\ 0, & \text{if } \textbf{Eval}\ [\![(Q_{m+1} \uparrow \binom{m\,+\,1}{1}) \downarrow \binom{Q_s}{0}]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m)\ =\ \mathit{false} \end{array} \right.,\quad \ldots, \\ \\ \quad e_p\ =\ \left\{ \begin{array}{ll} 1, & \text{if } \textbf{Eval}\ [\![(Q_p \uparrow \binom{m\,+\,1}{1}) \downarrow \binom{Q_s}{0}]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m)\ =\ \mathit{true}, \\ 0, & \text{if } \textbf{Eval}\ [\![(Q_p \uparrow \binom{m\,+\,1}{1}) \downarrow \binom{Q_s}{0}]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m)\ =\ \mathit{false} \end{array} \right., \\ \\ \quad x_0\ \in\ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho),\ldots, \\ \quad x_m\ \in\ \textbf{Eval}\ [\![Q_m]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_{m-1}), \\ \quad \textbf{Eval}\ [\![Q_w]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0,\ldots,\ x_m)\ =\ \mathit{true} \end{array} \right\},$$

$$= \left\{ \begin{array}{l} (\textbf{Eval}\ [\![V_0\ .\ a_0]\!]\ (\rho,\ x),\ f\ (x))\ |\ x\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho, \\ \quad f\ (x)\ =\ \{\textbf{Eval}\ [\![V_0\ .\ a_1]\!]\ (\rho,\ x,\ y)\ |\ y\ \in\ \textbf{Eval}\ [\![A]\!]\ \rho,\ \textbf{Eval}\ [\![V_1\ .\ a_0\ ==\ V_0\ .\ a_0]\!]\ (\rho,\ x,\ y)\} \end{array} \right\}$$

(where $\textbf{Eval}\ [\![A]\!]\ \rho\ =$

$$
\left\{
\begin{array}{l}
(\epsilon,\ e_{m+1}, \ldots,\ e_p,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \mid \\
\quad e_{m+1} \ = \left\{
\begin{array}{ll}
1, & \text{if } \textbf{Eval}\ [\![Q_{m+1} \uparrow \binom{m+1}{1}]\!] \\
 & \quad (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ [\![Q_{m+1} \uparrow \binom{m+1}{1}]\!] \\
 & \quad (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array}
\right. , \\
\quad \ldots, \\
\quad e_p \ = \left\{
\begin{array}{ll}
1, & \text{if } \textbf{Eval}\ [\![Q_p \uparrow \binom{m+1}{1}]\!] \\
 & \quad (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ [\![Q_p \uparrow \binom{m+1}{1}]\!] \\
 & \quad (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array}
\right. , \\
\quad x_0 \ \in \ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho), \ldots, \\
\quad x_m \ \in \ \textbf{Eval}\ [\![Q_m]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_{m-1}), \\
\quad \textbf{Eval}\ [\![Q_w]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m) \ = \ \mathit{true}
\end{array}
\right\}
$$

(By Lemma 3.2)

$$
= \left\{
\begin{array}{l}
(\textbf{Eval}\ [\![V_0\ .\ a_0]\!]\ (\rho,\ x),\ f\ (x)) \mid x \ \in \ \textbf{Eval}\ [\![A]\!]\ \rho, \\
\quad f\ (x) \ = \ \{\textbf{Eval}\ [\![V_0\ .\ a_1]\!]\ (\rho,\ x,\ y) \mid y \ \in \ \textbf{Eval}\ [\![A]\!]\ \rho,\ \textbf{Eval}\ [\![V_1\ .\ a_0 \ == \ V_0\ .\ a_0]\!]\ (\rho,\ x,\ y)\}
\end{array}
\right\}
$$

(where $\textbf{Eval}\ [\![A]\!]\ \rho \ =$

$$
\left\{
\begin{array}{l}
(\epsilon,\ e_{m+1}, \ldots,\ e_p,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \mid \\
\quad e_{m+1} \ = \left\{
\begin{array}{ll}
1, & \text{if } \textbf{Eval}\ [\![Q_{m+1}]\!] \\
 & \quad (\textbf{Eval}\ [\![Q']\!]\ \rho,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ [\![Q_{m+1}]\!] \\
 & \quad (\textbf{Eval}\ [\![Q']\!]\ \rho,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array}
\right. , \\
\quad \ldots, \\
\quad e_p \ = \left\{
\begin{array}{ll}
1, & \text{if } \textbf{Eval}\ [\![Q_p]\!] \\
 & \quad (\textbf{Eval}\ [\![Q']\!]\ \rho,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ [\![Q_p]\!] \\
 & \quad (\textbf{Eval}\ [\![Q']\!]\ \rho,\ \textbf{Eval}\ [\![Q_s]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array}
\right. , \\
\quad x_0 \ \in \ \textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho), \ldots, \\
\quad x_m \ \in \ \textbf{Eval}\ [\![Q_m]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_{m-1}), \\
\quad \textbf{Eval}\ [\![Q_w]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho,\ x_0, \ldots,\ x_m) \ = \ \mathit{true}
\end{array}
\right\})
$$

(By Lemma 3.2)

$$
= \left\{
\begin{array}{l}
(\delta,\ f\ (x)) \mid x \ = \ (\delta,\ v),\ x \ \in \ \textbf{Eval}\ [\![A]\!]\ \rho, \\
\quad f\ (x) \ = \ \{v' \mid y \ = \ (\delta',\ v'),\ y \ \in \ \textbf{Eval}\ [\![A]\!]\ \rho,\ \delta \ = \ \delta'\}
\end{array}
\right\}
$$

(where $\textbf{Eval}\ [\![A]\!]\ \rho \ =$

$$\left\{ \begin{array}{l}
(\epsilon,\ e_{m+1}, \ldots,\ e_p,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \mid \\
\quad e_{m+1} \ = \left\{ \begin{array}{ll}
1, & \text{if } \textbf{Eval}\ \llbracket Q_{m+1} \rrbracket \\
 & \quad (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ \llbracket Q_{m+1} \rrbracket \\
 & \quad (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array} \right. , \\
\quad \ldots, \\
\quad e_p \ = \left\{ \begin{array}{ll}
1, & \text{if } \textbf{Eval}\ \llbracket Q_p \rrbracket \\
 & \quad (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ \llbracket Q_p \rrbracket \\
 & \quad (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array} \right. , \\
\quad x_0 \ \in \ \textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho), \ldots, \\
\quad x_m \ \in \ \textbf{Eval}\ \llbracket Q_m \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_{m-1}), \\
\quad \textbf{Eval}\ \llbracket Q_w \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m) \ = \ \mathit{true}
\end{array} \right\})$$

$$= \left\{ \begin{array}{l}
((\epsilon,\ e_{m+1}, \ldots,\ e_p),\ f\ (x)) \mid x \ \in \ \textbf{Eval}\ \llbracket A \rrbracket\ \rho, \\
\quad f\ (x) \ = \ \{ \begin{array}{l}
\textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ y_0, \ldots,\ y_m) \mid \\
(\epsilon,\ e'_{m+1}, \ldots,\ e'_p,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ y_0, \ldots,\ y_m)) \ \in \ \textbf{Eval}\ \llbracket A \rrbracket\ \rho, \\
(\epsilon,\ e_{m+1}, \ldots,\ e_p) \ = \ (\epsilon,\ e'_{m+1}, \ldots,\ e'_p)
\end{array} \}
\end{array} \right\}$$

(where $\textbf{Eval}\ \llbracket A \rrbracket\ \rho \ =$

$$\left\{ \begin{array}{l}
(\epsilon,\ e_{m+1}, \ldots,\ e_p,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \mid \\
\quad e_{m+1} \ = \left\{ \begin{array}{ll}
1, & \text{if } \textbf{Eval}\ \llbracket Q_{m+1} \rrbracket \\
 & \quad (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ \llbracket Q_{m+1} \rrbracket \\
 & \quad (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array} \right. , \\
\quad \ldots, \\
\quad e_p \ = \left\{ \begin{array}{ll}
1, & \text{if } \textbf{Eval}\ \llbracket Q_p \rrbracket \\
 & \quad (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ \llbracket Q_p \rrbracket \\
 & \quad (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array} \right. , \\
\quad x_0 \ \in \ \textbf{Eval}\ \llbracket Q_0 \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho), \ldots, \\
\quad x_m \ \in \ \textbf{Eval}\ \llbracket Q_m \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_{m-1}), \\
\quad \textbf{Eval}\ \llbracket Q_w \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m) \ = \ \mathit{true}
\end{array} \right\})$$

$$= \left\{ \begin{array}{l}
((\epsilon,\ e_{m+1}, \ldots,\ e_p),\ f\ (x)) \mid x \ \in \ \textbf{Eval}\ \llbracket A \rrbracket\ \rho, \\
\quad e_{m+1} \ = \left\{ \begin{array}{ll}
1, & \text{if } \textbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array} \right. , \\
\quad \ldots, \\
\quad e_p \ = \left\{ \begin{array}{ll}
1, & \text{if } \textbf{Eval}\ \llbracket Q_p \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{true}, \\
0, & \text{if } \textbf{Eval}\ \llbracket Q_p \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \ \mathit{false}
\end{array} \right. , \\
\quad f\ (x) \ = \ \{ \begin{array}{l}
\textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ y_0, \ldots,\ y_m) \mid \\
(\epsilon,\ e'_{m+1}, \ldots,\ e'_p,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ y_0, \ldots,\ y_m)) \ \in \ \textbf{Eval}\ \llbracket A \rrbracket\ \rho, \\
\textbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \\
\quad \textbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ y_0, \ldots,\ y_m)), \ldots, \\
\textbf{Eval}\ \llbracket Q_p \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ x_0, \ldots,\ x_m)) \ = \\
\quad \textbf{Eval}\ \llbracket Q_p \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ \textbf{Eval}\ \llbracket Q_s \rrbracket\ (\textbf{Eval}\ \llbracket Q' \rrbracket\ \rho,\ y_0, \ldots,\ y_m)), \ldots,
\end{array} \}
\end{array} \right.$$

$$(\text{where } \mathbf{Eval}\ [\![A]\!]\ \rho\ =\ \mathbf{Eval}\ \left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ Q_s\\ \texttt{from}\ Q_0,\ldots,\ Q_m\\ \texttt{where}\ Q_w\end{array}\right]\!\!\right]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho))$$

$$=\ \mathbf{Eval}\ \left[\!\!\left[\begin{array}{l}\texttt{select distinct}\ Q_s\\ \texttt{from}\ Q_0,\ldots,\ Q_m\\ \texttt{where}\ Q_w\\ \texttt{groupby}\ Q_{m+1},\ldots,\ Q_p\end{array}\right]\!\!\right]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)$$

Case: $Q\ \equiv\ \texttt{if}\ Q_0\ \texttt{then}\ Q_1\ \texttt{else}\ Q_2$

$$\begin{array}{l}\mathbf{Eval}\ [\![\mathbf{T}\ [\![\texttt{if}\ Q_0\ \texttt{then}\ Q_1\ \texttt{else}\ Q_2]\!]\ \texttt{!}\ Q']\!]\ \rho\\ \quad =\ \mathbf{Eval}\ [\![\mathbf{con}\ (\mathbf{T}\ [\![Q_0]\!],\ \mathbf{T}\ [\![Q_1]\!],\ \mathbf{T}\ [\![Q_2]\!])\ \texttt{!}\ Q']\!]\ \rho\\ \quad =\ \mathbf{Eval}\ [\![\texttt{if}\ (\mathbf{T}\ [\![Q_0]\!]\ \texttt{?}\ Q')\ \texttt{then}\ (\mathbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q')\ \texttt{else}\ (\mathbf{T}\ [\![Q_2]\!]\ \texttt{!}\ Q')]\!]\ \rho\\ \quad =\ \begin{cases}\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho, & \text{if } \mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_0]\!]\ \texttt{?}\ Q']\!]\ \rho\ =\ \mathit{true},\\ \mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_2]\!]\ \texttt{!}\ Q']\!]\ \rho, & \text{if } \mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_0]\!]\ \texttt{?}\ Q']\!]\ \rho\ =\ \mathit{false}\end{cases}\\ \quad =\ \begin{cases}\mathbf{Eval}\ [\![Q_1]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho), & \text{if } \mathbf{Eval}\ [\![Q_0]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)\ =\ \mathit{true},\\ \mathbf{Eval}\ [\![Q_2]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho), & \text{if } \mathbf{Eval}\ [\![Q_0]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)\ =\ \mathit{false}\end{cases}\quad \text{(by induction)}\\ \quad =\ \mathbf{Eval}\ [\![\texttt{if}\ Q_0\ \texttt{then}\ Q_1\ \texttt{else}\ Q_2]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)\end{array}$$

Case: $Q\ \equiv\ (Q_0)$

$$\begin{array}{rcl}\mathbf{Eval}\ [\![\mathbf{T}\ [\![(Q_0)]\!]\ \texttt{!}\ Q']\!]\ \rho & = & \mathbf{Eval}\ [\![(\mathbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho\\ & = & (\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho)\\ & = & (\mathbf{Eval}\ [\![Q_0]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho))\quad \text{(by induction)}\\ & = & \mathbf{Eval}\ [\![(Q_0)]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)\end{array}$$

Case: $Q\ \equiv\ Q_0\ \texttt{==}\ Q_1$ (similarly for $!=, <, >, \leq, \geq$)

$$\begin{array}{l}\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_0\ \texttt{==}\ Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\\ \quad =\ \mathbf{Eval}\ [\![\mathbf{eq}\ \oplus\ \langle\mathbf{T}\ [\![Q_0]\!],\ \mathbf{T}\ [\![Q_1]\!]\rangle\ \texttt{?}\ Q']\!]\ \rho\\ \quad =\ \mathbf{Eval}\ [\![\mathbf{eq}\ \texttt{?}\ (\langle\mathbf{T}\ [\![Q_0]\!],\ \mathbf{T}\ [\![Q_1]\!]\rangle\ \texttt{!}\ Q')]\!]\ \rho\\ \quad =\ \mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q'\ \texttt{==}\ \mathbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\\ \quad =\ (\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho)\ =\ (\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho)\\ \quad =\ (\mathbf{Eval}\ [\![Q_0]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho))\ =\ (\mathbf{Eval}\ [\![Q_1]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho))\quad \text{(by induction)}\\ \quad =\ \mathbf{Eval}\ [\![Q_0\ \texttt{==}\ Q_1]\!]\ (\mathbf{Eval}\ [\![Q']\!]\ \rho)\end{array}$$

Case: $Q\ \equiv\ Q_0\ \texttt{in}\ Q_1$

$$\begin{array}{l}\mathbf{Eval}\ [\![\mathbf{T}\ [\![Q_0\ \texttt{in}\ Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\\ \quad =\ \mathbf{Eval}\ [\![\mathbf{exists}\ (\mathbf{eq})\ \oplus\ \langle\mathbf{T}\ [\![Q_0]\!],\ \mathbf{T}\ [\![Q_1]\!]\rangle\ \texttt{?}\ Q']\!]\ \rho\\ \quad =\ \mathbf{Eval}\ [\![\mathbf{exists}\ (\mathbf{eq})\ \texttt{?}\ (\langle\mathbf{T}\ [\![Q_0]\!],\ \mathbf{T}\ [\![Q_1]\!]\rangle\ \texttt{!}\ Q')]\!]\ \rho\end{array}$$

$$
\begin{aligned}
&= \textbf{Eval } [\![\texttt{exists}\ (\textbf{eq})\ \texttt{?} \ll \textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q',\ \textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q' \gg]\!]\ \rho \\
&= \textbf{Eval } [\![\texttt{exists}\ (\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q')\ :\ \textbf{eq}\ \texttt{?} \ll \textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q' \uparrow \binom{1}{0},\ V_0 \gg]\!]\ \rho \\
&= \textbf{Eval } [\![\texttt{exists}\ (\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q')\ :\ ((\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \binom{1}{0})\ ==\ V_0]\!]\ \rho \\
&= \exists\ x\ \in\ \textbf{Eval } [\![\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval } [\![(\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \binom{1}{0}\ ==\ V_0]\!]\ (\rho,\ x)\ =\ \mathit{true}) \\
&= \exists\ x\ \in\ \textbf{Eval } [\![\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval } [\![(\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \binom{1}{0}]\!]\ (\rho,\ x)\ =\ \textbf{Eval } [\![V_0]\!]\ (\rho,\ x)) \\
&= \exists\ x\ \in\ \textbf{Eval } [\![\textbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval } [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho\ =\ x)\ \ \text{(By Lemma 3.1)} \\
&= \exists\ x\ \in\ \textbf{Eval } [\![Q_1]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho)\ ((\textbf{Eval } [\![Q_0]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho))\ =\ x)\ \ \text{(by induction)} \\
&= (\textbf{Eval } [\![Q_0]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho))\ \in\ (\textbf{Eval } [\![Q_1]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho)) \\
&= \textbf{Eval } [\![Q_0\ \texttt{in}\ Q_1]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q\ \equiv\ \texttt{not}\ (Q_0)$

$$
\begin{aligned}
&\textbf{Eval } [\![\textbf{T}\ [\![\texttt{not}\ (Q_0)]\!]\ \texttt{!}\ Q']\!]\ \rho \\
&\quad = \textbf{Eval } [\![\sim\ (\textbf{T}\ [\![Q_0]\!])\ \texttt{?}\ Q']\!]\ \rho \\
&\quad = \textbf{Eval } [\![\texttt{not}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{?}\ Q')]\!]\ \rho \\
&\quad = \neg\ (\textbf{Eval } [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{?}\ Q']\!])\ \rho \\
&\quad = \neg\ (\textbf{Eval } [\![Q_0]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho))\ \ \text{(by induction)} \\
&\quad = \textbf{Eval } [\![\texttt{not}\ (Q_0)]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q\ \equiv\ Q_0\ \texttt{and}\ Q_1$ (similarly for `or`)

$$
\begin{aligned}
\textbf{Eval } [\![\textbf{T}\ [\![Q_0\ \texttt{and}\ Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ &=\ \textbf{Eval } [\![\textbf{T}\ [\![Q_0]\!]\ \&\ \textbf{T}\ [\![Q_1]\!]\ \texttt{?}\ Q']\!]\ \rho \\
&=\ \textbf{Eval } [\![(\textbf{T}\ [\![Q_0]\!]\ \texttt{?}\ Q')]\!]\ \rho\ \wedge\ \textbf{Eval } [\![(\textbf{T}\ [\![Q_1]\!]\ \texttt{?}\ Q')]\!]\ \rho \\
&=\ (\textbf{Eval } [\![Q_0]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho))\ \wedge\ (\textbf{Eval } [\![Q_1]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho))\ \ \text{(by induction)} \\
&=\ \textbf{Eval } [\![Q_0\ \texttt{and}\ Q_1]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q\ \equiv\ \texttt{for\_all}\ Q_0\ :\ Q_1$ (similarly for `exists`)

$$
\begin{aligned}
&\textbf{Eval } [\![\textbf{T}\ [\![\texttt{for\_all}\ Q_0\ :\ Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho \\
&\ = \textbf{Eval } [\![\textbf{forall}\ (\textbf{T}\ [\![Q_1]\!])\ \oplus\ \langle \textbf{id},\ \textbf{T}\ [\![Q_0]\!] \rangle\ \texttt{?}\ Q']\!]\ \rho \\
&\ = \textbf{Eval } [\![\textbf{forall}\ (\textbf{T}\ [\![Q_1]\!])\ \texttt{?}\ (\langle \textbf{id},\ \textbf{T}\ [\![Q_0]\!] \rangle\ \texttt{!}\ Q')]\!]\ \rho \\
&\ = \textbf{Eval } [\![\textbf{forall}\ (\textbf{T}\ [\![Q_1]\!])\ \texttt{?} \ll Q',\ \textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q' \gg]\!]\ \rho \\
&\ = \textbf{Eval } [\![\texttt{for\_all}\ (\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q')\ :\ (\textbf{T}\ [\![Q_1]\!]\ \texttt{?} \ll Q' \uparrow \binom{1}{0},\ V_0 \gg)]\!]\ \rho \\
&\ = \forall\ x\ \in\ \textbf{Eval } [\![\textbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval } [\![\textbf{T}\ [\![Q_1]\!]\ \texttt{?} \ll Q' \uparrow \binom{1}{0},\ V_0 \gg]\!]\ (\rho,\ x)\ =\ \mathit{true}) \\
&\ = \forall\ x\ \in\ (\textbf{Eval } [\![Q_0]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho))\ (\textbf{Eval } [\![Q_1]\!]\ (\textbf{Eval } [\![\ll Q' \uparrow \binom{1}{0},\ V_0 \gg]\!]\ (\rho,\ x)))\ =\ \mathit{true}) \\
&\qquad \text{(by induction)} \\
&\ = \forall\ x\ \in\ (\textbf{Eval } [\![Q_0]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho))\ (\textbf{Eval } [\![Q_1]\!]\ (\textbf{Eval } [\![Q' \uparrow \binom{1}{0}]\!]\ (\rho,\ x),\ \textbf{Eval } [\![V_0]\!]\ (\rho,\ x))\ =\ \mathit{true}) \\
&\ = \forall\ x\ \in\ (\textbf{Eval } [\![Q_0]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho))\ (\textbf{Eval } [\![Q_1]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho,\ x)\ =\ \mathit{true}) \\
&\ = \textbf{Eval } [\![\texttt{for\_all}\ Q_0\ :\ Q_1]\!]\ (\textbf{Eval } [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q\ \equiv\ \texttt{exists}\ (Q_0)$

$$
\textbf{Eval } [\![\textbf{T}\ [\![\texttt{exists}\ (Q_0)]\!]\ \texttt{!}\ Q']\!]\ \rho\ =\ \textbf{Eval } [\![\textbf{exists}\ (\texttt{K}_p\ (\texttt{T}))\ \oplus\ \langle \textbf{id},\ \textbf{T}\ [\![Q_0]\!] \rangle\ \texttt{?}\ Q']\!]\ \rho
$$

$$
\begin{aligned}
&= \textbf{Eval}\ [\![\textbf{exists}\,(\texttt{K}_p\ (\texttt{T}))\ \texttt{?}\ (\langle \textbf{id},\ \textbf{T}\,[\![Q_0]\!]\rangle\ \texttt{!}\ Q')]\!]\ \rho \\
&= \textbf{Eval}\ [\![\textbf{exists}\,(\texttt{K}_p\ (\texttt{T}))\ \texttt{?} \ll Q',\ \textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q' \gg]\!]\ \rho \\
&= \textbf{Eval}\ [\![\texttt{exists}\,(\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q')\ :\ (\texttt{K}_p\ (\texttt{T})\ \texttt{?} \ll Q' \uparrow \tbinom{1}{0},\ V_0 \gg)]\!]\ \rho \\
&= \exists\, x\ \in\ \textbf{Eval}\ [\![\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho \\
&= \exists\, x\ \in\ (\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho))\quad \text{(by induction)} \\
&= \textbf{Eval}\ [\![\texttt{exists}\,(Q_0)]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q\ \equiv\ \texttt{unique}\,(Q_0)$

$$
\begin{aligned}
\textbf{Eval}\ [\![\textbf{T}\,[\![\texttt{unique}\,(Q_0)]\!]\ \texttt{!}\ Q']\!]\ \rho\ &= \textbf{Eval}\ [\![\texttt{C}_p\ (\textbf{eq},\ \texttt{1})\ \oplus\ (\overline{\texttt{count}}\circ \textbf{T}\,[\![Q_0]\!])\ \texttt{?}\ Q']\!]\ \rho \\
&= \textbf{Eval}\ [\![\texttt{C}_p\ (\textbf{eq},\ \texttt{1})\ \texttt{?}\ (\overline{\texttt{count}}\ \texttt{!}\ (\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q'))]\!]\ \rho \\
&= \textbf{Eval}\ [\![(\overline{\texttt{count}}\ \texttt{!}\ (\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q'))\ \texttt{==}\ \texttt{1}]\!]\ \rho \\
&= \textbf{Eval}\ [\![\texttt{count}\ (\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q')\ \texttt{==}\ \texttt{1}]\!]\ \rho \\
&= \textbf{Eval}\ [\![\texttt{count}\ (\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q')]\!]\ \rho\ =\ 1 \\
&= |\textbf{Eval}\ [\![\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho|\ =\ 1 \\
&= |\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)|\ =\ 1 \\
&= \textbf{Eval}\ [\![\texttt{unique}\,(Q_0)]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q\ \equiv\ Q_0\ \texttt{==}\ \texttt{some}\,(Q_1)$ (and similarly for $!=, <, >, \leq, \geq$)

$$
\begin{aligned}
&\textbf{Eval}\ [\![\textbf{T}\,[\![Q_0\ \texttt{==}\ \texttt{some}\,(Q_1)]\!]\ \texttt{!}\ Q']\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\textbf{exists}\,(\textbf{eq})\ \oplus\ \langle \textbf{T}\,[\![Q_0]\!],\ \textbf{T}\,[\![Q_1]\!]\rangle\ \texttt{?}\ Q']\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\textbf{exists}\,(\textbf{eq})\ \texttt{?}\ (\langle \textbf{T}\,[\![Q_0]\!],\ \textbf{T}\,[\![Q_1]\!]\rangle\ \texttt{!}\ Q')]\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\textbf{exists}\,(\textbf{eq})\ \texttt{?} \ll \textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q',\ \textbf{T}\,[\![Q_1]\!]\ \texttt{!}\ Q' \gg]\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\texttt{exists}\,(\textbf{T}\,[\![Q_1]\!]\ \texttt{!}\ Q')\ :\ \textbf{eq}\ \texttt{?} \ll \textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q' \uparrow \tbinom{1}{0},\ V_0 \gg]\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\texttt{exists}\,(\textbf{T}\,[\![Q_1]\!]\ \texttt{!}\ Q')\ :\ ((\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \tbinom{1}{0})\ \texttt{==}\ V_0]\!]\ \rho \\
&\quad= \exists\, x\ \in\ \textbf{Eval}\ [\![\textbf{T}\,[\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval}\ [\![((\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \tbinom{1}{0})\ \texttt{==}\ V_0]\!]\ (\rho,\ x)\ =\ \mathit{true}) \\
&\quad= \exists\, x\ \in\ \textbf{Eval}\ [\![\textbf{T}\,[\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval}\ [\![(\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \tbinom{1}{0}]\!]\ (\rho,\ x)\ =\ \textbf{Eval}\ [\![V_0]\!]\ (\rho,\ x)) \\
&\quad= \exists\, x\ \in\ \textbf{Eval}\ [\![\textbf{T}\,[\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval}\ [\![\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho\ =\ x)\quad \text{(By Lemma 3.1)} \\
&\quad= \exists\, x\ \in\ \textbf{Eval}\ [\![Q_1]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)\ ((\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho))\ =\ x)\quad \text{(by induction)} \\
&\quad= \textbf{Eval}\ [\![Q_0\ \texttt{==}\ \texttt{some}\,(Q_1)]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}
$$

Case: $Q\ \equiv\ Q_0\ \texttt{==}\ \texttt{all}\,(Q_1)$ (and similarly for $!=, <, >, \leq, \geq$)

$$
\begin{aligned}
&\textbf{Eval}\ [\![\textbf{T}\,[\![Q_0\ \texttt{==}\ \texttt{all}\,(Q_1)]\!]\ \texttt{!}\ Q']\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\textbf{forall}\,(\textbf{eq})\ \oplus\ \langle \textbf{T}\,[\![Q_0]\!],\ \textbf{T}\,[\![Q_1]\!]\rangle\ \texttt{?}\ Q']\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\textbf{forall}\,(\textbf{eq})\ \texttt{?}\ (\langle \textbf{T}\,[\![Q_0]\!],\ \textbf{T}\,[\![Q_1]\!]\rangle\ \texttt{!}\ Q')]\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\textbf{forall}\,(\textbf{eq})\ \texttt{?} \ll \textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q',\ \textbf{T}\,[\![Q_1]\!]\ \texttt{!}\ Q' \gg]\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\texttt{for\_all}\,(\textbf{T}\,[\![Q_1]\!]\ \texttt{!}\ Q')\ :\ \textbf{eq}\ \texttt{?} \ll \textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q' \uparrow \tbinom{1}{0},\ V_0 \gg]\!]\ \rho \\
&\quad= \textbf{Eval}\ [\![\texttt{for\_all}\,(\textbf{T}\,[\![Q_1]\!]\ \texttt{!}\ Q')\ :\ ((\textbf{T}\,[\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \tbinom{1}{0})\ \texttt{==}\ V_0]\!]\ \rho
\end{aligned}
$$

$$
\begin{aligned}
&= \forall\, x \in \textbf{Eval}\ [\![\mathbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval}\ [\![((\mathbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \binom{1}{0}) \ == \ V_0]\!]\ (\rho,\ x)\ =\ \mathit{true})\\
&= \forall\, x \in \textbf{Eval}\ [\![\mathbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval}\ [\![(\mathbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q') \uparrow \binom{1}{0}]\!]\ (\rho,\ x)\ =\ \textbf{Eval}\ [\![V_0]\!]\ (\rho,\ x))\\
&= \forall\, x \in \textbf{Eval}\ [\![\mathbf{T}\ [\![Q_1]\!]\ \texttt{!}\ Q']\!]\ \rho\ (\textbf{Eval}\ [\![\mathbf{T}\ [\![Q_0]\!]\ \texttt{!}\ Q']\!]\ \rho\ =\ x)\ \ \text{(By Lemma 3.1)}\\
&= \forall\, x \in \textbf{Eval}\ [\![Q_1]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)\ ((\textbf{Eval}\ [\![Q_0]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho))\ =\ x)\ \ \text{(by induction)}\\
&= \textbf{Eval}\ [\![Q_0\ ==\ \texttt{all}\ (Q_1)]\!]\ (\textbf{Eval}\ [\![Q']\!]\ \rho)
\end{aligned}
$$

□

Corollary 4.1 (Weak Correctness) *For any non-Boolean OQL_{deB}* wfq *Q,*

$$\textbf{Eval}\ [\![\mathbf{T}\ [\![Q]\!]\ \texttt{!}\ \diamond]\!]\ \epsilon\ =\ \textbf{Eval}\ [\![Q]\!]\ \epsilon,$$

and for any Boolean OQL_{deB} wfq *Q,*

$$\textbf{Eval}\ [\![\mathbf{T}\ [\![Q]\!]\ \texttt{?}\ \diamond]\!]\ \epsilon\ =\ \textbf{Eval}\ [\![Q]\!]\ \epsilon.$$

Proof: Established trivially by setting $Q' = \diamond$and $\rho\ =\ \epsilon$ in Theorem 4.1. □

5 Related Work

We are not the first to suggest that a combinator-style for queries facilitates optimizer formalization and makes optimization more efficient. [EL91] consider the benefits of an FP-style ([Bac78]) query language from the perspective of optimization, but by proposing an FP-style query *language*. But combinator-style languages are difficult for users to master and thus ill-suited as query languages. [BG84] consider the benefits of FP-style languages with respect to formalization but provide no means of mapping query expressions into their FP-style algebra. We believe the translation technique described in this paper provides insight as to how this can be accomplished. [BF79] and [BK90] both propose an FP-style language to be used as an *internal* algebra of queries manipulated by an optimizer. Like [BG84], neither [BF79] nor [BK90] describe translation techniques to map queries expressed in query languages with variables into their formalism.

Since Turner's seminal paper [Tur79], combinators have been used within functional language compilers as internal representations of λ-expressions. The purpose of this is to make evaluation by graph reduction more efficient, as λ-expressions with free variables force unnecessary copying of potentially large function bodies [FH88]. Approaches to combinator translations can be classified according to whether the combinator set is *fixed* or *variable*. *Fixed* combinator sets consist of a finite set of combinators that are used as the target for all λ-expression translations. The best known of the fixed sets of combinators are the **SKI** combinator set, first introduced by Schönfinkel [Sch24]. It has been shown that this small set is sufficient to translate all of λ-calculus (in fact **I** is superfluous), but the size of the resulting code is too large to be of practical use [JRSB85]. Variations of the **SKI** combinator sets add additional, redundant combinators (e.g. **B** and **Y**) to reduce the size of the translated code. Curien proposed a set of Category Theory-inspired Combinators ([Cur93]) which he used to provide an alternative semantics for the λ-calculus. Of all of the combinator translations we found described in the literature, KOLA most closely resembles Curien's combinator set, but adjusted to account for sets, and avoiding the overly powerful combinators (**App** and Λ) that are expressive but difficult to optimize prior to their application to arguments.

λ-lifting [Joh85] and supercombinators [Hug84] are translation techniques that use variable sets of combinators. These techniques construct new combinators during each translation. The goal of this technique is to keep the number of combinators in the result small (the combinators generated tend to be fairly complicated). Thus, the goal of removing free variables from λ-expressions to make evaluation by graph reduction more efficient is reached, without an explosion in the size of the resulting code.

Despite the wide-spread use of supercombinators in functional language compilers, we settled on a fixed set of combinators for KOLA for the following reasons:

- query optimization (which relies on a set of known rewrite rules) must reference a known (i.e. fixed) set of operators.
- query optimization can tolerate query representations that are larger (within reason) than the queries themselves because queries tend to be small (compared with functional programs for example).

6 Conclusions

In this paper, we have proposed a technique for translating queries expressed in standard query languages into a variable-free (combinator) form. Our work is motivated by the difficulties that variables introduce into optimization, both in terms of an optimizer's efficiency and in terms of its formalization. We have described the technique by showing an example translator that maps queries written in a variant of OQL [Cat93] into queries written in KOLA [Che95a]; our combinator-based algebra. We have also presented proofs that establish the correctness of our translator. The proofs rely on a denotational semantics for our OQL variant and establish that translation produces a KOLA query whose OQL equivalent has the same semantics as the original query.

The significance of our work lies in its implications for the design of efficient and simply formalized query optimizers. With respect to formalization, we have used the Larch [GHG+92] specification tool LSL to formally specify KOLA and its associated rule language, and have used these specifications within the Larch theorem prover LP to verify the correctness of over 300 transformation rules. (For details of this work see [Che95a]). We were not the first to recognize the potential benefits of a combinator-based internal query algebra with respect to efficiency and formalization of query optimizers. But by the introduction of a translation technique to map queries into their combinator forms, these benefits become realizable in practice.

References

[Bac78] J. W. Backus. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641, August 1978.

[BF79] Peter Buneman and Robert E. Frankel. FQL – a functional query language. In *Proc. ACM SIGMOD Int'l Conference on Management of Data*, 1979.

[BG84] A. Bossi and C. Ghezzi. Using FP as a query language for relational data-bases. *Computer Languages*, 9(1):25–37, 1984.

[BK90] Catriel Beeri and Yoram Kornatzky. Algebraic optimization of object-oriented query languages. In S. Abiteboul and P. C. Kanellakis, editors, *Proceedings of the Third International Conference on Database Theory*, number 470 in Lecture Notes in Computer Science, pages 72–88, Paris, France, December 1990. EATCS, Springer-Verlag.

[BTBN91] Val Breazu-Tannen, Peter Buneman, and Shamim Naqvi. Structural recursion as a query language. In Paris Kanellakis and Joachim W. Schmidt, editors, *Bulk Types & Persistent Data: The Third International Workshop on Database Programming Languages*, pages 9–19, Nafplion, Greece, August 1991. Morgan Kaufmann Publishers, Inc.

[Cat93] R.G.G. Cattell, editor. *The Object Database Standard: ODMG-93*. Morgan-Kaufman, 1993.

[Che95a] Mitch Cherniack. Form(ers) over function(s): The KOLA query algebra. Technical report, Brown University Department of Computer Science, May 1995. In preparation.

[Che95b] Mitch Cherniack. A Larch specification of KOLA. Available at http://www.cs.brown.edu/software/cokokola/kola.tar.Z, 1995.

[Che96] Mitch Cherniack. An OQL $\rightarrow$ KOLA translator. Available at http://www.cs.brown.edu/software/cokokola/oqltrans.tar.Z, 1996.

[Cur93] Pierre-Louis Curien. *Categorical Combinators, Sequential Algorithms, and Functional Programming*. Birkhäuser, 1993.

[CZ96] Mitch Cherniack and Stanley B. Zdonik. Rule languages and internal algebras for rule-based optimizers. In *Proc. ACM SIGMOD Int'l Conference on Management of Data*, Montréal, Québec, Canada, June 1996.

[CZLK96] Mitch Cherniack, Stan Zdonik, Joon-Suk Lee, and Kee-Eung Kim. Grouping rules the COKO-KOLA way: It's the real thing. Submitted to SIGMOD '97, 1996.

[CZN95] Mitch Cherniack, Stanley B. Zdonik, and Marian H. Nodine. To form a more perfect union (intersection, difference). In *Proc. 5th Int'l Workshop on Database Programming Languages*, Gubbio, Italy, September 1995. Springer-Verlag.

[DeB72] N.G. DeBruijn. Lambda calculus notation with nameless dummies. *Indagationes Mathematicae*, 34:381–392, 1972.

[EL91] Martin Erwig and Udo W. Lipeck. A functional DBPL revealing high level optimizations. In Paris Kanellakis and Joachim W. Schmidt, editors, *Bulk Types & Persistent Data: The Third International Workshop on Database Programming Languages*, pages 306–, Nafplion, Greece, August 1991. Morgan Kaufmann Publishers, Inc.

[FH88] Anthony J. Field and Peter G. Harrison. *Functional Programming*. International Computer Science Press. Addison-Wesley, 1988.

[GHG+92] J.V. Guttag, J.J. Hornung, S.J. Garland, K.D. Jones, A. Modet, and J.M. Wing. *Larch: Languages and Tools for Formal Specifications*. Springer-Verlag, 1992.

[Hug84] R. J. M. Hughes. *The design and implementation of programming languages*. PhD thesis, University of Oxford, 1984.

[Joh85] T. Johnsson. Lambda lifting: transforming programs to recursive equations. In *Conference on Functional Programming Languages and Computer Architecture*, LNCS. Springer Verlag, 1985.

[JRSB85] M.S. Joy, V.J. Rayward-Smith, and F.W. Burton. Efficient combinator code. *Computer Languages*, 10:211–224, 1985.

[Kim82] Won Kim. On optimizing an SQL-like nested query. *ACM Transactions on Database Systems*, 7(3):443–469, September 1982.

[LMS+93] Theodore W. Leung, Gail Mitchell, Bharathi Subramanian, Bennet Vance, Scott L. Vandenberg, and Stanley B. Zdonik. The AQUA data model and algebra. In *Proc. 4th Int'l Workshop on Database Programming Languages*, New York, New York, August 1993. Springer-Verlag.

[Sch24] M. Schönfinkel. Über die bausteine der mathematischen logik. *Math. Annalen*, 92:305–316, 1924.

[Tur79] D. A. Turner. A new implementation technique for applicative languages. *Software - Practice and Experience*, 9:31–49, 1979.

$$
\begin{aligned}
\mathbf{Eval}\ \llbracket c \rrbracket\ \rho &= \mathbf{Val}\ \llbracket c \rrbracket,\ (\text{where } \mathbf{Val}\ \llbracket \{\} \rrbracket = \oslash,\ \mathbf{Val}\ \llbracket \diamond \rrbracket = \epsilon,\ \mathbf{Val}\ \llbracket \mathtt{T} \rrbracket = \mathit{true} \text{ etc.})\\
\mathbf{Eval}\ \llbracket V_i \rrbracket\ (\epsilon,\ d_n, \ldots,\ d_0) &= d_i \ \text{ if } 0 \leq i \leq n\\
\mathbf{Eval}\ \llbracket \ll Q_0, \ldots,\ Q_m \gg \rrbracket\ \rho &= (\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho, \ldots,\ \mathbf{Eval}\ \llbracket Q_m \rrbracket\ \rho)\\
\mathbf{Eval}\ \llbracket Q\ .\ a_0 \rrbracket\ \rho &= \delta,\ \text{if } \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho = (\delta,\ v)).\\
\mathbf{Eval}\ \llbracket Q\ .\ a_1 \rrbracket\ \rho &= v,\ \text{if } \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho = (\delta,\ v)).\\
\mathbf{Eval}\ \llbracket Q \rightarrow \mathtt{att} \rrbracket\ \rho &= \mathit{att}\ (\mathbf{Eval}\ \llbracket Q \rrbracket\ \rho),\ \text{if } \mathit{att} \text{ is an attribute defined in the type of } Q.
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{Eval}\ \llbracket -\ Q \rrbracket\ \rho &= -\ (\mathbf{Eval}\ \llbracket Q \rrbracket\ \rho)\\
\mathbf{Eval}\ \llbracket \mathtt{abs}\ (Q) \rrbracket\ \rho &= |\mathbf{Eval}\ \llbracket Q \rrbracket\ \rho|\\
\mathbf{Eval}\ \llbracket Q_0\ +\ Q_1 \rrbracket\ \rho &= (\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho) + (\mathbf{Eval}\ \llbracket Q_1 \rrbracket\ \rho)\ \ (\text{sim. for } -,\ *,\ /,\ \mathtt{mod})\\
\mathbf{Eval}\ \llbracket Q_0\ \mathtt{intersect}\ Q_1 \rrbracket\ \rho &= (\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho) \cap (\mathbf{Eval}\ \llbracket Q_1 \rrbracket\ \rho)\ \ (\text{sim. for } \mathtt{union},\ \mathtt{except})
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{Eval}\ \llbracket \mathtt{set}\ (Q_0, \ldots, Q_m) \rrbracket\ \rho &= \{\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho, \ldots, \mathbf{Eval}\ \llbracket Q_m \rrbracket\ \rho\}\\
\mathbf{Eval}\ \llbracket \mathtt{element}\ (Q) \rrbracket\ \rho &= x,\ \text{if } \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho = \{x\}.\\
\mathbf{Eval}\ \llbracket \mathtt{flatten}\ (Q) \rrbracket\ \rho &= \{y \mid \exists\ x \in \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho\ (y \in x)\}\\
\mathbf{Eval}\ \llbracket \mathtt{count}\ (Q) \rrbracket\ \rho &= |\mathbf{Eval}\ \llbracket Q \rrbracket\ \rho|\\
\mathbf{Eval}\ \llbracket \mathtt{sum}\ (Q) \rrbracket\ \rho &= \textstyle\sum_{x \in \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho} x\\
\mathbf{Eval}\ \llbracket \mathtt{avg}\ (Q) \rrbracket\ \rho &= \begin{cases} \mathbf{Eval}\ \llbracket \mathtt{sum}\ (Q) \rrbracket\ \rho \div \mathbf{Eval}\ \llbracket \mathtt{count}\ (Q) \rrbracket\ \rho, & \text{if } |\mathbf{Eval}\ \llbracket Q \rrbracket\ \rho| > 0.\\ 0, & \text{if } |\mathbf{Eval}\ \llbracket Q \rrbracket\ \rho| = 0. \end{cases}\\
\mathbf{Eval}\ \llbracket \mathtt{min}\ (Q) \rrbracket\ \rho &= x,\ \text{if } x \in \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho \text{ and } \forall\ y \in \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho\ (x \leq y).\\
\mathbf{Eval}\ \llbracket \mathtt{max}\ (Q) \rrbracket\ \rho &= x,\ \text{if } x \in \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho \text{ and } \forall\ y \in \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho\ (x \geq y).
\end{aligned}
$$

$$
\mathbf{Eval} \left\llbracket \begin{array}{l} \mathtt{select\ distinct}\ Q_s\\ \mathtt{from}\ Q_0, \ldots, Q_m\\ \mathtt{where}\ Q_w \end{array} \right\rrbracket \rho = \left\{ \begin{array}{l} \mathbf{Eval}\ \llbracket Q_s \rrbracket\ (\rho,\ x_0, \ldots,\ x_m) \mid\\ \quad x_0 \in \mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho, \ldots,\\ \quad x_m \in \mathbf{Eval}\ \llbracket Q_m \rrbracket\ (\rho,\ x_0, \ldots,\ x_{m-1}),\\ \quad \mathbf{Eval}\ \llbracket Q_w \rrbracket\ (\rho,\ x_0, \ldots,\ x_m) = \mathit{true} \end{array} \right\}
$$

$$
\mathbf{Eval} \left\llbracket \begin{array}{l} \mathtt{select\ distinct}\ Q_s\\ \mathtt{from}\ Q_0, \ldots, Q_m\\ \mathtt{where}\ Q_w\\ \mathtt{groupby}\ Q_{m+1}, \ldots, Q_p \end{array} \right\rrbracket \rho = \left\{ \begin{array}{l} (\epsilon,\ e_{m+1}, \ldots, e_p, f\ (x)) \mid\ x \in R,\\ \quad e_{m+1} = \begin{cases} 1, & \text{if } \mathbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\rho, x) = \mathit{true},\\ 0, & \text{if } \mathbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\rho, x) = \mathit{false} \end{cases}, \ldots,\\ \\ \quad e_p = \begin{cases} 1, & \text{if } \mathbf{Eval}\ \llbracket Q_p \rrbracket\ (\rho, x) = \mathit{true},\\ 0, & \text{if } \mathbf{Eval}\ \llbracket Q_p \rrbracket\ (\rho, x) = \mathit{false} \end{cases}\\ \\ \quad f(x) = \left\{ \begin{array}{l} y \in R \mid\\ \quad \mathbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\rho, x) = \mathbf{Eval}\ \llbracket Q_{m+1} \rrbracket\ (\rho, y),\\ \quad \ldots,\\ \quad \mathbf{Eval}\ \llbracket Q_p \rrbracket\ (\rho, x) = \mathbf{Eval}\ \llbracket Q_p \rrbracket\ (\rho, y) \end{array} \right\} \end{array} \right\}
$$

$$
(\text{where } R = \mathbf{Eval} \left\llbracket \begin{array}{l} \mathtt{select\ distinct}\ Q_s\\ \mathtt{from}\ Q_0, \ldots,\ Q_m\\ \mathtt{where}\ Q_w \end{array} \right\rrbracket \rho)
$$

$$
\begin{aligned}
\mathbf{Eval}\ \llbracket \mathtt{if}\ Q_0\ \mathtt{then}\ Q_1\ \mathtt{else}\ Q_2 \rrbracket\ \rho &= \begin{cases} \mathbf{Eval}\ \llbracket Q_1 \rrbracket\ \rho, & \text{if } \mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho = \mathit{true}\\ \mathbf{Eval}\ \llbracket Q_2 \rrbracket\ \rho, & \text{if } \mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho = \mathit{false} \end{cases}\\
\mathbf{Eval}\ \llbracket (Q) \rrbracket\ \rho &= (\mathbf{Eval}\ \llbracket Q \rrbracket\ \rho)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{Eval}\ \llbracket Q_0\ ==\ Q_1 \rrbracket\ \rho &= (\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho) = (\mathbf{Eval}\ \llbracket Q_1 \rrbracket\ \rho)\ \ (\text{sim. for } !=, <, >, \geq, \leq)\\
\mathbf{Eval}\ \llbracket Q_0\ \mathtt{in}\ Q_1 \rrbracket\ \rho &= (\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho) \in (\mathbf{Eval}\ \llbracket Q_1 \rrbracket\ \rho)\\
\mathbf{Eval}\ \llbracket \mathtt{not}\ (Q) \rrbracket\ \rho &= \neg\ (\mathbf{Eval}\ \llbracket Q \rrbracket\ \rho)\\
\mathbf{Eval}\ \llbracket Q_0\ \mathtt{and}\ Q_1 \rrbracket\ \rho &= (\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho) \wedge (\mathbf{Eval}\ \llbracket Q_1 \rrbracket\ \rho)\ \ (\text{sim. for } \mathtt{or})
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{Eval}\ \llbracket \mathtt{for\_all}\ Q_0\ :\ Q_1 \rrbracket\ \rho &= \forall\ x \in \mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho\ (\mathbf{Eval}\ \llbracket Q_1 \rrbracket\ (\rho, x) = \mathit{true})\\
\mathbf{Eval}\ \llbracket \mathtt{exists}\ Q_0\ :\ Q_1 \rrbracket\ \rho &= \exists\ x \in \mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho\ (\mathbf{Eval}\ \llbracket Q_1 \rrbracket\ (\rho, x) = \mathit{true})\\
\mathbf{Eval}\ \llbracket \mathtt{exists}\ (Q) \rrbracket\ \rho &= \exists\ x \in \mathbf{Eval}\ \llbracket Q \rrbracket\ \rho\\
\mathbf{Eval}\ \llbracket \mathtt{unique}\ (Q) \rrbracket\ \rho &= |\mathbf{Eval}\ \llbracket Q \rrbracket\ \rho| = 1\\
\mathbf{Eval}\ \llbracket Q_0 == \mathtt{some}\ (Q_1) \rrbracket\ \rho &= \exists\ x \in \mathbf{Eval}\ \llbracket Q_1 \rrbracket\ \rho\ (\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho = x)\ \ (\text{sim. for } !=, <, >, \geq, \leq)\\
\mathbf{Eval}\ \llbracket Q_0 == \mathtt{all}\ (Q_1) \rrbracket\ \rho &= \forall\ x \in \mathbf{Eval}\ \llbracket Q_1 \rrbracket\ \rho\ (\mathbf{Eval}\ \llbracket Q_0 \rrbracket\ \rho = x)\ \ (\text{sim. for } !=, <, >, \geq, \leq)
\end{aligned}
$$

Figure 4: A Denotational Semantics of OQL_{deB}

Q	$Q \uparrow \binom{k}{j}$	$Q \downarrow \binom{Q'}{j}$
c V_i	c $\begin{cases} V_{i+k}, & \text{if } i \geq j \\ V_i, & \text{if } i < j \end{cases}$	c $\begin{cases} V_{i-1}, & \text{if } i > j \\ Q', & \text{if } i = j \\ V_i, & \text{if } i < j \end{cases}$
$\ll Q_0, \ldots,\ Q_m \gg$ $Q\ .\ a_0$ $Q\ .\ a_1$ $Q \rightarrow\ \mathtt{att}$	$\ll Q_0 \uparrow \binom{k}{j}, \ldots,\ Q_m \uparrow \binom{k}{j} \gg$ $(Q \uparrow \binom{k}{j})\ .\ a_0$ $(Q \uparrow \binom{k}{j})\ .\ a_1$ $(Q \uparrow \binom{k}{j}) \rightarrow\ \mathtt{att}$	$\ll Q_0 \downarrow \binom{Q'}{j}, \ldots,\ Q_m \downarrow \binom{Q'}{j} \gg$ $(Q \downarrow \binom{Q'}{j})\ .\ a_0$ $(Q \downarrow \binom{Q'}{j})\ .\ a_1$ $(Q \downarrow \binom{Q'}{j}) \rightarrow\ \mathtt{att}$
$-Q$ $\mathtt{abs}\ (Q)$ $Q_0\ \ \mathit{ArithOp}\ \ Q_1$ $Q_0\ \ \mathit{SetOp}\ \ Q_1$	$-\ (Q \uparrow \binom{k}{j})$ $\mathtt{abs}\ (Q \uparrow \binom{k}{j})$ $Q_0 \uparrow \binom{k}{j}\ \ \mathit{ArithOp}\ \ Q_1 \uparrow \binom{k}{j}$ $Q_0 \uparrow \binom{k}{j}\ \ \mathit{SetOp}\ \ Q_1 \uparrow \binom{k}{j}$	$-\ (Q \downarrow \binom{Q'}{j})$ $\mathtt{abs}\ (Q \downarrow \binom{Q'}{j})$ $Q_0 \downarrow \binom{Q'}{j}\ \ \mathit{ArithOp}\ \ Q_1 \downarrow \binom{Q'}{j}$ $Q_0 \downarrow \binom{Q'}{j}\ \ \mathit{SetOp}\ \ Q_1 \downarrow \binom{Q'}{j}$
$\mathtt{set}\ (Q_0, \ldots,\ Q_n)$ $\mathtt{element}\ (Q)$ $\mathtt{flatten}\ (Q)$ $\mathit{Agg}\ (Q)$	$\mathtt{set}\ (Q_0 \uparrow \binom{k}{j}, \ldots,\ Q_n \uparrow \binom{k}{j})$ $\mathtt{element}\ (Q \uparrow \binom{k}{j})$ $\mathtt{flatten}\ (Q \uparrow \binom{k}{j})$ $\mathit{Agg}\ (Q \uparrow \binom{k}{j})$	$\mathtt{set}\ (Q_0 \downarrow \binom{Q'}{j}, \ldots,\ Q_n \downarrow \binom{Q'}{j})$ $\mathtt{element}\ (Q \downarrow \binom{Q'}{j})$ $\mathtt{flatten}\ (Q \downarrow \binom{Q'}{j})$ $\mathit{Agg}\ (Q \downarrow \binom{Q'}{j})$
$\mathtt{select\ distinct}\ Q_s$ $\mathtt{from}\ Q_0, \ldots,\ Q_m$ $\mathtt{where}\ Q_w$ $\mathtt{select\ distinct}\ Q_s$ $\mathtt{from}\ Q_0, \ldots,\ Q_m$ $\mathtt{where}\ Q_w$ $\mathtt{groupby}\ Q_{m+1}, \ldots,\ Q_n$	$\mathtt{select\ distinct}\ Q_s \uparrow \binom{k}{j+m+1}$ $\mathtt{from}\ Q_0 \uparrow \binom{k}{j}, \ldots,\ Q_m \uparrow \binom{k}{j+m}$ $\mathtt{where}\ Q_w \uparrow \binom{k}{j+m+1}$ $\mathtt{select\ distinct}\ Q_s \uparrow \binom{k}{j+m+1}$ $\mathtt{from}\ Q_0 \uparrow \binom{k}{j}, \ldots,\ Q_m \uparrow \binom{k}{j+m}$ $\mathtt{where}\ Q_w \uparrow \binom{k}{j+m+1}$ $\mathtt{groupby}\ Q_{m+1} \uparrow \binom{k}{j+1}, \ldots,\ Q_n \uparrow \binom{k}{j+1}$	$\mathtt{select\ distinct}\ Q_s \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1}$ $\mathtt{from}\ Q_0 \downarrow \binom{Q'}{j}, \ldots,\ Q_m \downarrow \binom{Q' \uparrow \binom{m}{0}}{j+m}$ $\mathtt{where}\ Q_w \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1}$ $\mathtt{select\ distinct}\ Q_s \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1}$ $\mathtt{from}\ Q_0 \downarrow \binom{Q'}{j}, \ldots,\ Q_m \downarrow \binom{Q' \uparrow \binom{m}{0}}{j+m}$ $\mathtt{where}\ Q_w \downarrow \binom{Q' \uparrow \binom{m+1}{0}}{j+m+1}$ $\mathtt{groupby}\ Q_{m+1} \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1}, \ldots,\ Q_n \downarrow \binom{Q' \uparrow \binom{1}{0}}{j+1}$
$\mathtt{if}\ Q_0\ \mathtt{then}\ Q_1\ \mathtt{else}\ Q_2$ (Q)	$\mathtt{if}\ (Q_0 \uparrow \binom{k}{j})\ \mathtt{then}(\ Q_1 \uparrow \binom{k}{j})\ \mathtt{else}\ (Q_2 \uparrow \binom{k}{j})$ $(Q \uparrow \binom{k}{j})$	$\mathtt{if}\ (Q_0 \downarrow \binom{Q'}{j})\ \mathtt{then}\ (Q_1 \downarrow \binom{Q'}{j})\ \mathtt{else}\ (Q_2 \downarrow \binom{Q'}{j})$ $(Q \downarrow \binom{Q'}{j})$
$Q_0\ \ \mathit{CompOp}\ \ Q_1$ $Q_0\ \mathtt{in}\ Q_1$ $\mathtt{not}\ (Q)$ $Q_0\ \ \mathit{BoolOp}\ \ Q_1$	$Q_0 \uparrow \binom{k}{j}\ \ \mathit{CompOp}\ \ Q_1 \uparrow \binom{k}{j}$ $(Q_0 \uparrow \binom{k}{j})\ \mathtt{in}\ (Q_1 \uparrow \binom{k}{j})$ $\mathtt{not}\ (Q \uparrow \binom{k}{j})$ $(Q_0 \uparrow \binom{k}{j})\ \ \mathit{BoolOp}\ \ (Q_1 \uparrow \binom{k}{j})$	$Q_0 \downarrow \binom{Q'}{j}\ \ \mathit{CompOp}\ \ Q_1 \downarrow \binom{Q'}{j}$ $(Q_0 \downarrow \binom{Q'}{j})\ \mathtt{in}\ (Q_1 \downarrow \binom{Q'}{j})$ $\mathtt{not}\ (Q \downarrow \binom{Q'}{j})$ $(Q_0 \downarrow \binom{Q'}{j})\ \ \mathit{BoolOp}\ \ (Q_1 \downarrow \binom{Q'}{j})$
$\mathtt{for\_all}\ Q_0\ \mathtt{:}\ Q_1$ $\mathtt{exists}\ Q_0\ \mathtt{:}\ Q_1$ $\mathtt{exists}\ (Q)$ $\mathtt{unique}\ (Q)$ $Q_0\ \ \mathit{CompOp}\ \ \mathtt{some}\ Q_1$ $Q_0\ \ \mathit{CompOp}\ \ \mathtt{all}\ Q_1$	$\mathtt{for\_all}\ (Q_0 \uparrow \binom{k}{j})\ \ \mathtt{:}\ \ (Q_1 \uparrow \binom{k}{j})$ $\mathtt{exists}\ (Q_0 \uparrow \binom{k}{j})\ \ \mathtt{:}\ \ (Q_1 \uparrow \binom{k}{j})$ $\mathtt{exists}\ (Q \uparrow \binom{k}{j})$ $\mathtt{unique}\ (Q \uparrow \binom{k}{j})$ $(Q_0 \uparrow \binom{k}{j})\ \ \mathit{CompOp}\ \ \mathtt{some}\ (Q_1 \uparrow \binom{k}{j})$ $(Q_0 \uparrow \binom{k}{j})\ \ \mathit{CompOp}\ \ \mathtt{all}\ (Q_1 \uparrow \binom{k}{j})$	$\mathtt{for\_all}\ (Q_0 \downarrow \binom{Q'}{j})\ \ \mathtt{:}\ \ (Q_1 \downarrow \binom{Q'}{j})$ $\mathtt{exists}\ (Q_0 \downarrow \binom{Q'}{j})\ \ \mathtt{:}\ \ (Q_1 \downarrow \binom{Q'}{j})$ $\mathtt{exists}\ (Q \downarrow \binom{Q'}{j})$ $\mathtt{unique}\ (Q \downarrow \binom{Q'}{j})$ $(Q_0 \downarrow \binom{Q'}{j})\ \ \mathit{CompOp}\ \ \mathtt{some}\ (Q_1 \downarrow \binom{Q'}{j})$ $(Q_0 \downarrow \binom{Q'}{j})\ \ \mathit{CompOp}\ \ \mathtt{all}\ (Q_1 \downarrow \binom{Q'}{j})$

Figure 5: Formal Definitions of *Abstraction* $(Q \uparrow \binom{k}{j})$ and *Application* $(Q \downarrow \binom{Q'}{j})$

KOLA Combinator	OQL Equivalent	
Identity	$\mathbf{id} \ ! \ x$	x
Projection-1	$\pi_1 \ ! \ x$	$x \ . \ a_0$
Projection-2	$\pi_2 \ ! \ x$	$x \ . \ a_1$
Arithmetic	$\mathbf{add} \ ! \ \ll i, \ j \gg$	$i \ + \ j$ (similarly for **sub** (-), **mul** (*), **div** (/), **mod** (`mod`))
Inverse	$\mathbf{inverse} \ ! \ i$	$-i$
Absolute Value	$\mathbf{abs} \ ! \ i$	`abs` (i)
Insertion	$\mathbf{ins} \ ! \ \ll x, \ A \gg$	`set` (x) `union` A
Singleton	$\mathbf{set} \ ! \ x$	`set` (x)
Element Extraction	$\mathbf{choose} \ ! \ A$	`element` (A)
Set Flatten	$\mathbf{flat} \ ! \ A$	`flatten` (A)
Object Attributes-1	$\mathtt{att} \ ! \ x$	$x \rightarrow \mathtt{att}$
General Comparison	$\mathbf{eq} \ ? \ \ll x, \ y \gg$	$x \ == \ y$ (similarly for **neq** ($!=$))
Integer Comparison	$\mathbf{lt} \ ? \ \ll i, \ j \gg$	$i \ < \ j$ (similarly for **gt** ($>$), **leq** ($\leq$), **geq** ($\geq$))
Object Attributes-2	$\mathtt{att} \ ? \ x$	$x \rightarrow \mathtt{att}$
Composition	$(f \ \circ \ g) \ ! \ x$	$f \ ! \ (g \ ! \ x)$
Function Pairing	$\langle f, \ g \rangle \ ! \ x$	$\ll f \ ! \ x, \ g \ ! \ x \gg$
Constant Functions	$\mathtt{K}_f \ (\mathtt{c}) \ ! \ x$	`c` (for OQL$_{deB}$ constant `c`)
Curried Functions	$\mathtt{C}_f \ (f, \ x) \ ! \ y$	$f \ ! \ \ll x, \ y \gg$
Function Pairs	$(f \ \times \ g) \ ! \ \ll x, \ y \gg$	$\ll f \ ! \ x, \ g \ ! \ y \gg$
Conditionals	$\mathbf{con} \ (p, \ f, \ g) \ ! \ x$	`if` $(p \ ? \ x)$ `then` $(f \ ! \ x)$ `else` $(g \ ! \ x)$
Combination	$(p \ \oplus \ f) \ ? \ x$	$p \ ? \ (f \ ! \ x)$
Conjunction	$(p \ \& \ q) \ ? \ x$	$(p \ ? \ x)$ `and` $(q \ ? \ x)$
Disjunction	$(p \mid q) \ ? \ x$	$(p \ ? \ x)$ `or` $(q \ ? \ x)$
Negation	$(\sim \ p) \ ? \ x$	`not` $(p \ ? \ x)$
Constant Predicates	$\mathtt{K}_p \ (\mathtt{b}) \ ? \ x$	`b` (for OQL$_{deB}$ boolean constant `b`)
Curried Predicates	$\mathtt{C}_p \ (p, \ x) \ ? \ y$	$p \ ? \ \ll x, \ y \gg$
Iteration	$\mathbf{iterate} \ (p, \ f) \ ! \ A$	`select distinct` $f \ ! \ V_0$ `from` A `where` $p \ ? \ V_0$
Pair-with Iteration	$\mathbf{iter} \ (p, \ f) \ ! \ \ll x, \ A \gg$	`select distinct` $f \ ! \ \ll x \uparrow \binom{1}{0}, \ V_0 \gg$ `from` A `where` $p \ ? \ \ll x \uparrow \binom{1}{0}, \ V_0 \gg$
Intersection	$\mathbf{int} \ (eq) \ ! \ \ll A, \ B \gg$	A^B_{eq} `intersect` B^A_{eq} (similarly for **uni** (`union`), **dif** (`except`)) $(X^Y_{eq} \ =$ `select distinct` V_0 `from` X `where (exists` $(Y \uparrow \binom{1}{0})$: $eq \ ? \ \ll V_1, \ V_0 \gg)$
Partial Aggregates	$\mathbf{bound} \ (p) \ ! \ A$	`element` (`select distinct` V_0 `from` A `where (for_all` $(A \uparrow \binom{1}{0})$: $p \ ? \ \ll V_1, \ V_0 \gg)$)
Total Aggregates	$\overline{\mathtt{count}} \ ! \ \mathrm{A}$ $\overline{\mathtt{sum}} \ ! \ \mathrm{A}$ $\overline{\mathtt{avg}} \ ! \ \mathrm{A}$	`count` (A) `sum` (A) `avg` (A)
Grouping	$\mathbf{grp} \ (p, \ f, \ g) \ ! \ A$	`select distinct` $f \ ! \ \ll V_0,$ (`select distinct` $g \ ! \ V_0$ `from` $A \uparrow \binom{1}{0}$ `where` $p \ ? \ \ll V_1, \ V_0 \gg$) $\gg$ `from` A
$\exists$ Quantification-1	$\mathbf{exists} \ (p) \ ? \ A$	`exists` A : $(p \ ? \ V_0)$
$\forall$ Quantification-1	$\mathbf{forall} \ (p) \ ? \ A$	`for_all` A : $(p \ ? \ V_0)$
$\exists$ Quantification-2	$\mathbf{EX} \ (p) \ ? \ \ll x, \ A \gg$	`exists` A : $(p \ ? \ \ll x \uparrow \binom{1}{0}, \ V_0 \gg)$
$\forall$ Quantification-2	$\mathbf{FA} \ (p) \ ? \ \ll x, \ A \gg$	`for_all` A : $(p \ ? \ \ll x \uparrow \binom{1}{0}, \ V_0 \gg)$

Table 2: The Syntax and Semantics of KOLA

$$
\begin{aligned}
\mathbf{T}\llbracket c \rrbracket &= \begin{cases} \mathtt{K}_f\ (c), & \text{if } c \text{ is any constant of type other than } \mathit{Bool} \\ \mathtt{K}_p\ (c), & \text{if where } c \text{ is } \mathtt{T} \text{ or } \mathtt{F} \end{cases} \\
\mathbf{T}\llbracket V_i \rrbracket &= \pi_2 \circ {\pi_1}^i \ \ (\text{where } {\pi_1}^0 = \mathbf{id}.) \\
\mathbf{T}\llbracket \ll Q_0, \ldots, Q_m \gg \rrbracket &= \langle \ldots \langle \mathbf{T}\llbracket Q_0 \rrbracket, \mathbf{T}\llbracket Q_1 \rrbracket \rangle, \ldots, \mathbf{T}\llbracket Q_m \rrbracket \rangle \\
\mathbf{T}\llbracket Q\ .\ a_0 \rrbracket &= \pi_1 \circ \mathbf{T}\llbracket Q \rrbracket \\
\mathbf{T}\llbracket Q\ .\ a_1 \rrbracket &= \pi_2 \circ \mathbf{T}\llbracket Q \rrbracket \\
\mathbf{T}\llbracket Q\ .\ \mathtt{att} \rrbracket &= \begin{cases} \mathtt{att} \circ \mathbf{T}\llbracket Q \rrbracket, & \text{if } \mathtt{att} \text{ is an object attribute of any type but } \mathit{Bool}. \\ \mathtt{att} \oplus \mathbf{T}\llbracket Q \rrbracket, & \text{if } \mathtt{att} \text{ is an object attribute of type } \mathit{Bool}. \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{T}\llbracket -\ Q \rrbracket &= \mathbf{inverse} \circ \mathbf{T}\llbracket Q \rrbracket \\
\mathbf{T}\llbracket \mathtt{abs}\ (Q) \rrbracket &= \mathbf{abs} \circ \mathbf{T}\llbracket Q \rrbracket \\
\mathbf{T}\llbracket Q_0 + Q_1 \rrbracket &= \mathbf{add} \circ \langle \mathbf{T}\llbracket Q_0 \rrbracket, \mathbf{T}\llbracket Q_1 \rrbracket \rangle \ (\text{sim. for - } (\mathbf{sub}),\ *\ (\mathbf{mul}),\ \ (\mathbf{div}),\ \mathtt{mod}\ (\mathbf{mod})) \\
\mathbf{T}\llbracket Q_0\ \mathtt{intersect}\ Q_1 \rrbracket &= \mathbf{int}\ (\mathbf{eq}) \circ \langle \mathbf{T}\llbracket Q_0 \rrbracket, \mathbf{T}\llbracket Q_1 \rrbracket \rangle \ (\text{sim. for } \mathtt{union}\ (\mathbf{uni}\ (\mathbf{eq})),\ \mathtt{except}\ (\mathbf{dif}\ (\mathbf{eq}))) \\
\mathbf{T}\llbracket \mathtt{set}\ (Q_0, \ldots, Q_n) \rrbracket &= \mathbf{ins} \circ \langle \mathbf{T}\llbracket Q_0 \rrbracket, \mathbf{ins} \circ \ldots \circ \mathbf{ins} \circ \langle \mathbf{T}\llbracket Q_n \rrbracket, \mathtt{K}_f\ (\oslash) \rangle \ldots \rangle \\
\mathbf{T}\llbracket \mathtt{element}\ (Q) \rrbracket &= \mathbf{choose} \circ \mathbf{T}\llbracket Q \rrbracket \\
\mathbf{T}\llbracket \mathtt{flatten}\ (Q) \rrbracket &= \mathbf{flat} \circ \mathbf{T}\llbracket Q \rrbracket \\
\mathbf{T}\llbracket \mathtt{count}\ (Q) \rrbracket &= \overline{\mathtt{count}} \circ \mathbf{T}\llbracket Q \rrbracket \ (\text{similarly for } \mathtt{sum}\ (\overline{\mathtt{sum}}) \text{ and } \mathtt{avg}\ (\overline{\mathtt{avg}}).) \\
\mathbf{T}\llbracket \mathtt{min}\ (Q) \rrbracket &= \mathbf{bound}\ (\mathbf{leq}) \circ \mathbf{T}\llbracket Q \rrbracket \ (\text{similarly for } \mathtt{max}\ (\mathbf{geq}).)
\end{aligned}
$$

$$
\mathbf{T}\left\llbracket \begin{array}{l} \mathtt{select\ distinct}\ Q_s \\ \mathtt{from}\ Q_0, \ldots, Q_m \\ \mathtt{where}\ Q_w \end{array} \right\rrbracket = \begin{array}{l} \mathbf{iterate}\ (\mathbf{T}\llbracket Q_w \rrbracket,\ \mathbf{T}\llbracket Q_s \rrbracket) \circ \\ \ \mathbf{flat} \circ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathbf{T}\llbracket Q_m \rrbracket \rangle) \circ \\ \ \ \ldots \circ \\ \ \ \mathbf{flat} \circ \mathbf{iterate}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{iter}\ (\mathtt{K}_p\ (\mathtt{T}),\ \mathbf{id}) \circ \langle \mathbf{id},\ \mathbf{T}\llbracket Q_0 \rrbracket \rangle) \circ \\ \ \ \ \mathbf{set} \end{array}
$$

$$
\mathbf{T}\left\llbracket \begin{array}{l} \mathtt{select\ distinct}\ Q_s \\ \mathtt{from}\ Q_0, \ldots, Q_m \\ \mathtt{where}\ Q_w \\ \mathtt{groupby}\ Q_{m+1}, \ldots, Q_p \end{array} \right\rrbracket = \begin{array}{l} \mathbf{grp}\ (\mathbf{eq} \oplus (\pi_1 \times \pi_1),\ \pi_1 \times \mathbf{id},\ \pi_2) \circ \\ \quad \mathbf{T}\left\llbracket \begin{array}{l} \mathtt{select\ distinct} \ll \diamond,\ f_{m+1},\ \ldots,\ f_p,\ Q_s \gg \\ \mathtt{from}\ Q_0, \ldots, Q_m \\ \mathtt{where}\ Q_w \end{array} \right\rrbracket \\ \quad (\text{where } f_i = \mathtt{if}\ ((Q_i \uparrow \binom{m+1}{1}) \downarrow \binom{Q_s}{0})\ \mathtt{then\ 1\ else\ 0}) \end{array}
$$

$$
\begin{aligned}
\mathbf{T}\llbracket \mathit{if}\ Q_0\ \mathit{then}\ Q_1\ \mathit{else}\ Q_2 \rrbracket &= \mathbf{con}\ (\mathbf{T}\llbracket Q_0 \rrbracket,\ \mathbf{T}\llbracket Q_1 \rrbracket,\ \mathbf{T}\llbracket Q_2 \rrbracket) \\
\mathbf{T}\llbracket (Q) \rrbracket &= \mathbf{T}\llbracket Q \rrbracket
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{T}\llbracket Q_0\ \mathtt{==}\ Q_1 \rrbracket &= \mathbf{eq} \oplus \langle \mathbf{T}\llbracket Q_0 \rrbracket, \mathbf{T}\llbracket Q_1 \rrbracket \rangle \\
&\quad (\text{sim. for } \mathtt{!=}\ (\mathbf{neq}),\ <\ (\mathbf{lt}),\ >\ (\mathbf{gt}),\ <=\ (\mathbf{leq}),\ >=\ (\mathbf{geq}),\ \mathtt{in}\ (\mathbf{EX}\ (\mathbf{eq}))) \\
\mathbf{T}\llbracket \mathtt{not}\ Q \rrbracket &= \sim \mathbf{T}\llbracket Q \rrbracket \\
\mathbf{T}\llbracket Q_0\ \mathtt{and}\ Q_1 \rrbracket &= \mathbf{T}\llbracket Q_0 \rrbracket\ \&\ \mathbf{T}\llbracket Q_1 \rrbracket \\
\mathbf{T}\llbracket Q_0\ \mathtt{or}\ Q_1 \rrbracket &= \mathbf{T}\llbracket Q_0 \rrbracket \mid \mathbf{T}\llbracket Q_1 \rrbracket
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{T}\llbracket \mathtt{for\_all}\ Q_0 : Q_1 \rrbracket &= \mathbf{FA}\ (\mathbf{T}\llbracket Q_1 \rrbracket) \oplus \langle \mathbf{id},\ \mathbf{T}\llbracket Q_0 \rrbracket \rangle \\
\mathbf{T}\llbracket \mathtt{exists}\ Q_0 : Q_1 \rrbracket &= \mathbf{EX}\ (\mathbf{T}\llbracket Q_1 \rrbracket) \oplus \langle \mathbf{id},\ \mathbf{T}\llbracket Q_0 \rrbracket \rangle \\
\mathbf{T}\llbracket \mathtt{exists}\ (Q) \rrbracket &= \mathbf{EX}\ (\mathbf{T}\llbracket \mathtt{K}_p\ (\mathtt{T}) \rrbracket) \oplus \langle \mathbf{id},\ \mathbf{T}\llbracket Q \rrbracket \rangle \\
\mathbf{T}\llbracket \mathtt{unique}\ (Q) \rrbracket &= \mathtt{C}_p\ (\mathbf{eq},\ 1) \oplus (\overline{\mathtt{count}} \circ \mathbf{T}\llbracket Q \rrbracket) \\
\mathbf{T}\llbracket Q_0\ \mathtt{==}\ \mathtt{some}\ (Q_1) \rrbracket &= \mathbf{EX}\ (\mathbf{eq}) \oplus \langle \mathbf{T}\llbracket Q_0 \rrbracket, \mathbf{T}\llbracket Q_1 \rrbracket \rangle \\
&\quad (\text{sim. for } \mathtt{!=}\ (\mathbf{neq}),\ <\ (\mathbf{lt}),\ >\ (\mathbf{gt}),\ <=\ (\mathbf{leq}),\ >=\ (\mathbf{geq})) \\
\mathbf{T}\llbracket Q_0\ \mathtt{==}\ \mathtt{all}\ (Q_1) \rrbracket &= \mathbf{FA}\ (\mathbf{eq}) \oplus \langle \mathbf{T}\llbracket Q_0 \rrbracket, \mathbf{T}\llbracket Q_1 \rrbracket \rangle \\
&\quad (\text{sim. for } \mathtt{!=}\ (\mathbf{neq}),\ <\ (\mathbf{lt}),\ >\ (\mathbf{gt}),\ <=\ (\mathbf{leq}),\ >=\ (\mathbf{geq}))
\end{aligned}
$$

Figure 6: **T**: The OQL → KOLA Translation Function