Semantics Preservation

In this chapter, we state some proofs related to semantics-preservation. We first show that a set of semantics-preserving operation types is a semantics-preserving set of operation types. Then, we prove in detail that the operation type AddResultPredicate with parameter argument is semantics-preserving. Afterward, we sketch the proof that the remaining operation types are semantics preserving.

1 Set of Semantics-Preserving Operations

Proposition 1. Given an arbitrary, consistent set of operation types T such that each operation $t \in T$ is semantics-preserving, then T is a semantics-preserving set of operation types (see Definition 7.5).

Proof. In the following, we prove Proposition 1 for a set of two operation types where each of which targets only a single constituent. For sets of operation types that comprise more than two operation types, the proposition could be proved similarly. Furthermore, for operations types that target more than one constituent, the proposition could be proved similarly.

Table 1 defines items that will be used in the proof. Roughly speaking, the main idea of the proof is splitting a set of two operations, each from a different type, into two sets each of which comprises a single operation. Then, for each of the resulting sets with a single operation, we inspect the outcome of applying the set with a single operation to a source situation with respect to semantics preservation. Finally, the outcomes from the application of both sets are combined together. For the reader's convenience, Figure 1 helps to understand the idea of the proof. The reader is advised to refer to Figure 1 while reading through the proof.

Note that we consider the case when each of o_a and o_b does not result in an empty update set when evaluated against q_s as well as that each of o'_a and o'_b does not result in an empty update set when evaluated against q'_s (see Definition 7.5).

An analysis situation is defined as a tuple of functions (Definition 5.7). Hence, q_s has the form in Listing 1, where $1 \leq a < b \leq n \geq 2$ and the function f_i^s denotes the interpretation of the function name f_i in the situation q_s (for each i such that $1 \leq i \leq n$).

Listing 1: Situation q_s

$q_s = (f_1^s, ..., f_a^s, ..., f_b^s, ..., f_n^s)$

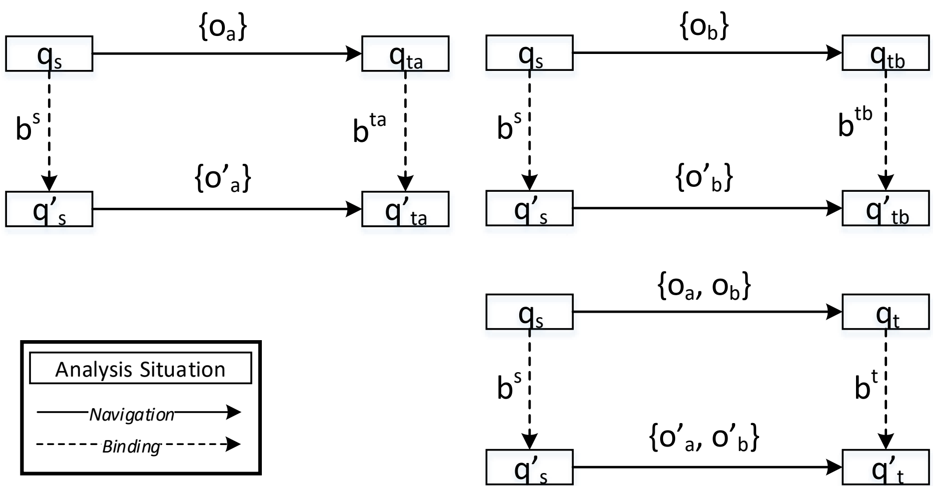

Figure 1: Ideas used in the proof

Table 1: Definitions of items used in the proof

Item	Definition
t_a	An arbitrary, semantics-preserving operation type
t_b	An arbitrary, semantics-preserving operation type different from t_a, where $\{t_a, t_b\}$ is consistent
T	A consistent set of operation types such that $T = \{t_a, t_b\}$
o_a	An arbitrary operation of the type t_a
o_b	An arbitrary operation of the type t_b
f_a	The function name targeted by o_a
f_b	The function name targeted by o_b such that $f_a \neq f_b$
O	A set of operations such that $O = \{o_a, o_b\}$
o'_a	An arbitrary, ground operation of the operation o_a
o'_b	An arbitrary, ground operation of the operation o_b
q_s	An arbitrary analysis situation
n	A navigation step that has the source situation q_s and the set of operations O
q_t	The target situation of n

An arbitrary tuple of bindings b^s for the situation q_s has a binding function for each function in that situation (Definition 5.8). Hence, b^s has the form in Listing 2, where π_i is a binding function for the function name f_i for each i such that $1 \leq i \leq n$. Note that b^s must: (i) bind all unbound parameters of q_s and bind only unbound parameters of q_s (refer to Section 7.2.1), (ii) bind each parameter to a value of the parameter's domain (first property in Definition 5.8), and (iii) satisfy that the situation resulting from the application of b^s to q_s fulfills the second property in Definition 5.8, related to dice levels and nodes.

Listing 2: Binding tuple b^s

$$b^s = (\pi_1, ..., \pi_a, ..., \pi_b, ..., \pi_n)$$

Let q'_s be the situation resulting from the application of the binding tuple b^s to the situation q_s. The situation q'_s has the form in Listing 3, where the function $f_i^{s'}$ denotes the interpretation of the function name f_i in the situation q'_s (for each i such that $1 \leq i \leq n$).

Listing 3: Situation q'_s

$$q'_s = \beta^{q_s}(b^s) = (f_1^{s'}, ..., f_a^{s'}, ..., f_b^{s'}, ..., f_n^{s'})$$

Let q_{t_a} be the situation resulting from evaluating and firing the operation o_a in the situation q_s.

The situation q_{t_a} has the form in Listing 4. Note that, in comparison to q_s (Listing 1), only the interpretation of the function name f_a changes in q_{t_a} since o_a targets the function name f_a.

Listing 4: Situation q_{t_a}

$$q_{t_a} = \delta^{q_s}(\epsilon^{q_s}(o_a)) = (f_1^s, ..., f_a^{t_a}, ..., f_b^s, ..., f_n^s)$$

Let q'_{t_a} (Listing 5) be the situation resulting from the evaluation and firing of o'_a in q'_s. Note that in comparison to q'_s (Listing 3), only the interpretation of the function name f_a changes in q'_{t_a} since o'_a targets the function name f_a.

Listing 5: Situation q'_{t_a}

$$q'_{t_a} = \delta^{q'_s}(\epsilon^{q'_s}(o'_a)) = (f_1^{s'}, ..., f_a^{t'_a}, ..., f_b^{s'}, ..., f_n^{s'})$$

Note that it is possible to find a unique binding tuple the satisfies semantics preservation if such a binding tuple exists (refer to the notes after Definition 7.6). In particular, the binding tuple has to bind all unbound parameters of the target situation and bind only unbound parameters of the target situation. This also implies that each binding function in the binding tuple binds all unbound parameters of the situation's function which the binding function targets, and binds only unbound parameters of the situation's function which the binding function targets.

Let b^{t_a} be the described unique binding tuple of q_{t_a} that satisfies semantics preservation for: (i) the situation q_s, (ii) the binding tuple b^s, (iii) the operation o_a, and (iv) the binding function π that results in the ground operation o'_a when applied to o_a (see Definition 7.6). In order for b^{t_a} to be the described unique binding tuple that preserves semantics, it must hold that: (i) $\delta^{q'_s}(\epsilon^{q'_s}(o'_a)) = \beta^{q_{t_a}}(b^{t_a})$, i.e., $q'_{t_a} = \beta^{q_{t_a}}(b^{t_a})$, (ii) b^{t_a} binds all unbound parameters of q_{t_a} and binds only unbound parameters of q_{t_a}, (iii) b^{t_a} binds each parameter to a value of its domain, and (iv) b^{t_a} preserves the second property in Definition 5.8, related to dice nodes and dice levels.

We claim that the desired binding tuple b^{t_a} has the form in Listing 6, which is identical to b^s (Listing 2) except for the binding function for the function name f_a, i.e., π'_a is a member of b^{t_a} instead of π_a in b^s. We show the correctness of that claim in the following.

Listing 6: Binding tuple b^{t_a}

$$b^{t_a} = (\pi_1, ..., \pi'_a, ..., \pi_b, ..., \pi_n)$$

We start by proving the point (i), i.e., $q'_{t_a} = \beta^{q_{t_a}}(b^{t_a})$. For each i such that $1 \leq i \leq n$ and $i \neq a$, applying π_i to f_i^s results in $f_i^{s'}$ (see Listings 1, 2, and 3). Hence, for each i such that $1 \leq i \leq n$ and $i \neq a$, in order to obtain the function $f_i^{s'}$ in q'_{t_a} from the function f_i^s in q_{t_a}, b^{t_a} must contain the respective binding function π_i (see Listings 4 and 5). Consequently, b^{t_a} has the form in Listing 6, where π'_a is a binding function that may be different from π_a in b^s and results $f_a^{t'_a}$ when applied to $f_a^{t_a}$.

Now we prove the points (ii) and (iii), i.e., b^{t_a} binds all unbound parameters of q_{t_a} and binds only unbound parameters of q_{t_a}, and binds each parameter to a value of the parameter's domain. We, first, require that π'_a binds all unbound parameters of $f_a^{t'_a}$ and binds only unbound parameters of $f_a^{t'_a}$, and binds each parameter in the domain of π'_a to a value of the parameter's domain. The remaining binding functions π_i, for each i such that $1 \leq i \leq n$ and $i \neq a$, also bind all unbound parameters of their respective functions and bind only unbound parameters of their respective functions, and bind each parameter in the domain of the binding function to a value of the parameter's domain. The reason behind that is that (i) these binding functions are identical to their counterparts in b^s and we required that b^s binds all unbound parameters and binds only unbound parameters, and binds each parameter to a value of the parameter's domain, and (ii) each function f_i^s in q_{t_a}, for each i such that $1 \leq i \leq n$ and $i \neq a$, is identical to its counterpart in q_s.

Point (iv) is assumed to be true since the application of b^{t_a} to q_{t_a} results in q'_{t_a} which preserves the second property in Definition 5.8 since we require in Section 6.2 that the application of operations to analysis situations results in analysis situations. Note also that point (iii) can be assumed to be true for the same reason.

As similar reasoning leads to the forms of q_{t_b}, q'_{t_b}, and b^{t_b}, which are shown in Listings 7, 8, and 9, respectively. Note that these forms in Listings 7, 8, and 9 are related to the operation o_b instead of the operation o_a.

Listing 7: Situation q_{t_b}

$$q_{t_b} = \delta^{q_s}(\epsilon^{q_s}(o_b)) = (f_1^s, ..., f_a^s, ..., f_b^{t_b}, ..., f_n^s)$$

Listing 8: Situation q'_{t_b}

$$q'_{t_b} = \delta^{q'_s}(\epsilon^{q'_s}(o'_b)) = (f_1^{s'}, ..., f_a^{s'}, ..., f_b^{t'_b}, ..., f_n^{s'})$$

Listing 9: Binding tuple b^{t_b}

$$b^{t_b} = (\pi_1, ..., \pi_a, ..., \pi'_b, ..., \pi_n)$$

The situation q_t, resulting from evaluating and firing $\{o_a, o_b\}$ in q_s, has the form in Listing 10. Note that the differences from q_s (Listing 1) are only the interpretations of the function names f_a and f_b, since the operation o_a targets the function name f_a and the operation o_b targets the function name f_b. Note also that the interpretation of the function name f_a (which is $f_a^{t_a}$) is identical in q_t and q_{t_a} (Listing 4). The reason behind that is that in both cases we evaluate and fire o_a (that targets only f_a in the source situation) against the same interpretation of f_a which is f_a^s since q_t and q_{t_a} share the same source situation q_s. Similarly, the interpretation of the function name f_b (which is $f_b^{t_b}$) is identical in q_t and q_{t_b} (Listing 7).

Listing 10: Situation q_t

$$q_t = \delta^{q_s}(\epsilon^{q_s}(\{o_a, o_b\})) = \{f_1^s, ..., f_a^{t_a}, ..., f_b^{t_b}, ..., f_n^s\}$$

In order to prove Proposition 1, we need to show that there exists a binding tuple x that is when applied to q_t results in the same as $\delta^{q'_s}(\epsilon^{q'_s}(\{o'_a, o'_b\}))$. We claim that this binding tuple x is the tuple b^t shown in Listing 11.

Listing 11: Binding tuple b^t

```
b^t = (π_1, ..., π'_a, ..., π'_b, ..., π_n)
```

In order to prove our claim, we need to prove two points: (I) b^t is a binding tuple of q_t, and (II) $\beta^{q_t}(b^t) = \delta^{q'_s}(\epsilon^{q'_s}(\{o'_a, o'_b\}))$, which we do in the following.

(I) b^t is a binding tuple of q_t. In order to prove that, we need to show that each binding function in b^t is a binding function for its respective function in q_t (which implies that each individual binding function binds all unbound parameters and only unbound parameters to values of the parameter's domain), and that the second property in Definition 5.8 is preserved.

We first need to show that π'_a is a binding function for $f_a^{t_a}$ and that π'_b is a binding function for $f_b^{t_b}$:

- b^{t_a} (Listing 6) is a binding tuple for q_{t_a} (Listing 4), which implies that π'_a is a binding function for $f_a^{t_a}$.
- b^{t_b} (Listing 9) is a binding tuple for q_{t_b} (Listing 7), which implies that π'_b is a binding function for $f_b^{t_b}$.

For each remaining π_i, where $1 \leq i \leq n$, $a \neq i$, and $b \neq i$, it holds that π_i is a binding function for f_i^s since: (i) the interpretation functions are identical to their counterparts in and q_s, (ii) the binding functions are identical to their counterparts in b^s, and (iii) we have established that b^s (Listing 2) is a binding tuple for q_s (Listing 1).

Note that the application of b^t to q_t can be assumed to result in an analysis situation that preserves the second property in Definition 5.8, if the next point ($\beta^{q_t}(b^t) = \delta^{q'_s}(\epsilon^{q'_s}(\{o'_a, o'_b\}))$) is correct, since the application of b^t to q_t, then, results in q'_t which preserves the second property in Definition 5.8 since we require in Section 6.2 that the application of operations to analysis situations results in analysis situations. Note also that the point that each binding function in b^t binds each parameter in the domain of the binding function to a value of the parameter's domain can also be assumed to be true for the same reason.

(II) $\beta^{q_t}(b^t) = \delta^{q'_s}(\epsilon^{q'_s}(\{o'_a, o'_b\}))$. In order to prove that, let us inspect the result of the application of the binding tuple b^t (Listing 11) to the situation q_t (Listing 10), and then compare the result with the situation q'_t (Listing 12) that results from evaluating and firing $\{o'_a, o'_b\}$ in q'_s.

Note that q'_t has the form in Listing 12 since o'_a and o'_b target f_a and f_b respectively, which means that only f_a and f_b may be different from q'_s and all other functions are identical in q'_s and q'_t. Note also that the interpretation of the function name f_a (which is $f_a^{t'_a}$) is identical in q'_t and q'_{t_a}.

The reason behind that is that in both cases we evaluate and fire o'_a (that targets only f_a in the source situation) against the same interpretation of f_a which is $f_a^{s'}$ since q'_t and q'_{t_a} share the same source situation q'_s. Similarly, the interpretation of the function name f_b (which is $f_b^{t'_b}$) is identical in q'_t and q'_{t_b}.

Listing 12: Situation q'_t

$$q'_t = \delta^{q'_s}(\epsilon^{q'_s}(\{o'_a, o'_b\})) = (f_1^{s'}, ..., f_a^{t'_a}, ..., f_b^{t'_b}, ..., f_n^{s'})$$

In the following, we use the notation for the application of a tuple of bindings to a situation to denote the application of a single binding function to its respective function in a situation (see Definition 5.9). In particular, given the function name f_x and the binding function π_x that corresponds to the function name f_x, the application of π_x to the function f_x^q of a situation q, denoted as $\beta^{f_x^q}(\pi_x)$, replaces the binding of each parameter $p \in \mathit{unbound}(f_x^q)$ in f_x^q with $\pi_x(p)$, if $\pi_x(p)$ is defined. Hence the application of the tuple of bindings b^t (Listing 11) to the analysis situation q_t (Listing 10), i.e., $\beta^{q_t}(b^t)$, can be written as in Listing 13.

Listing 13: The application $\beta^{q_t}(b^t)$

$$\beta^{q_t}(b^t) = (\beta^{f_1^s}(\pi_1), ..., \beta^{f_a^{t_a}}(\pi'_a), ..., \beta^{f_b^{t_b}}(\pi'_b), ..., \beta^{f_n^s}(\pi_n))$$

Recall that we want to prove that $\beta^{q_t}(b^t) = \delta^{q'_s}(\epsilon^{q'_s}(\{o'_a, o'_b\}))$, which can also be written as $\beta^{q_t}(b^t) = q'_t$. By comparing $\beta^{q_t}(b^t)$ in Listing 13 with q'_t in Listing 12, it follows that we need to prove the equalities shown in Listing 14.

Listing 14: Equalities to be proved

$$\beta^{f_1^s}(\pi_1) = f_1^{s'}, ..., \beta^{f_a^{t_a}}(\pi'_a) = f_a^{t'_a}, ..., \beta^{f_b^{t_b}}(\pi'_b) = f_b^{t'_b}, ..., \beta^{f_n^s}(\pi_n) = f_n^{s'}$$

We start with the equalities involving the function names f_a and f_b. Since we have shown that $\beta^{q_{t_a}}(b^{t_a}) = q'_{t_a}$, it follows from Listings 4, 5, and 6 that $\beta^{f_a^{t_a}}(\pi'_a) = f_a^{t'_a}$. Similarly, it follows from Listings 7, 8, and 9 that $\beta^{f_b^{t_b}}(\pi'_b) = f_b^{t'_b}$.

We now move to the remaining equalities that do not involve the function names f_a and f_b. Since we have established that $\beta^{q_s}(b^s) = q'_s$, it follows from Listings 1, 2, and 3 that $\beta^{f_i^s}(\pi_i) = f_i^{s'}$ where $1 \leq i \leq n$ and $a \neq i$ and $b \neq i$.

□

Case $f_a = f_b$ Note that we assumed that $f_a \neq f_b$ in Table 1. In case $f_a = f_b$, we briefly show that Proposition 1 holds in the following. If $f_a = f_b$, either o_a or o_b will produce an empty update set, or both of them will produce the same update set (since O is consistent because T is consistent). In the following, we show that T is semantics-preserving in each of the previous scenarios:

- When $\epsilon^{q_s}(\{o_a\}) = \emptyset$ or $\epsilon^{q_s}(\{o_b\}) = \emptyset$, then Definition 7.5 holds.

- When $\epsilon^{q_s}(\{o_a\}) = \epsilon^{q_s}(\{o_b\}) \neq \emptyset$, we have the following options:
  - When $\epsilon^{q'_s}(\{o'_a\}) = \emptyset$ or $\epsilon^{q'_s}(\{o'_b\}) = \emptyset$, then Definition 7.5 holds.
  - When $\epsilon^{q'_s}(\{o'_a\}) = \epsilon^{q'_s}(\{o'_b\}) \neq \emptyset$, then Definition 7.5 is satisfied since we know that t_a is semantics-preserving and effectively it holds that $O = \{o_a\}$ and that $O' = \{o'_a\}$ (or consider t_b, instead, which is semantics-preserving and effectively it holds that $O = \{o_b\}$ and that $O' = \{o'_b\}$).

2 Add Parameter Result Predicate

Proposition 2. The operation type $t = (\textsc{AddResultPredicate}, (P_{|B_{M_R}}, B_{M_R} \cup U), updSet_{\text{ADDRP}_1})$ (Table 6.2) is semantics-preserving.

Proof. Table 7 defines items that will be used in the proof. Roughly speaking, the main idea of the proof is checking the various possible scenarios for an operation of the type t and showing, for each scenario, that the operation type t is semantics-preserving. In this proof, we only consider the function name *filter* of an analysis situation since the operation type t targets only that function name, which means that only the interpretation function of the function name *filter* may be altered by an operation of the type t.

Note that given Definition 7.6, we say that the operation o of the type t is at the analysis graph level and that the ground operation $o' = \beta^o(\pi)$, of the operation o, is at the actual analysis level.

Table 2: Definitions of items used in the proof

Item	Definition
p	An arbitrary parameter $p \in P_{\vert B_{M_R}}$
c	An arbitrary value $c \in B_{M_R} \cup U$
o	A navigation operation $o = \textsc{AddResultPredicate}(p, c)$ of the type t, at the analysis graph level
c'	An arbitrary value $c' \in B_{M_R}$
o'	A ground operation $o' = \textsc{AddResultPredicate}(p, c')$ of the operation o, at the actual analysis level (note that if $c \neq ?$ then $c' = c$; see Definition 6.9 and Definition 6.10)
π	A binding function of o such that $\beta^o(\pi) = o'$ (note that if $c \neq ?$ then $\pi = \emptyset$ and if $c = ?$ then $\pi = \{2 \mapsto c'\}$)
q_1	An arbitrary analysis situation
b^1	An arbitrary binding tuple for q_1
q_1'	The situation resulting from the application of b^1 to q_1, i.e., $\beta^{q_1}(b^1) = q_1'$
q_2'	The situation resulting from the evaluation and firing of o' in q_1', i.e., $\delta^{q_1'}(\epsilon^{q_1'}(o')) = q_2'$
q_2	The situation resulting from the evaluation and firing of o in q_1, i.e., $\delta^{q_1}(\epsilon^{q_1}(o)) = q_2$

Definition 7.6 is not concerned with scenarios that result in empty update sets at the analysis graph level or at the actual analysis level. Hence, we do not need to consider the cases in which o or o' leads to empty update sets.

By inspecting the definition of the function $updSet_{\text{ADDRP}_1}$ in Table 6.2, we can rule out the cases in which o results in an empty update set when applied to q_1, which are shown in Listing 15.

Listing 15: Cases in which o results in an empty update set when applied to q_1

$p \in paras(\mathit{filter}^{q_1})$

$c \notin dom_{\mathfrak{p}}(p) \cup U$

Similarly, we can rule out the cases in which o' results in an empty update set when applied to q_1', which are shown in Listing 49.

Listing 16: Cases in which o' results in an empty update set when applied to q_1'

$p \in paras(\mathit{filter}^{q_1'})$

$c' \notin dom_{\mathfrak{p}}(p) \cup U$

By ruling out these cases, we end up with scenarios in which both o and o' result in a non-empty update set.

Based on the definition of a bindable set (Definition 5.5) as well as the definition of the function name *filter* (Definition 5.7), filter^{q_1} has the form shown in Listing 88, where: (i) $n \geq 0$, (ii) $0 \leq k \leq m$, (iii) $\{c_1, ..., c_n\}$ is a set where $\{c_1, ..., c_n\} \subseteq B_{M_R}$, (iv) $\{p_1, ..., p_k\}$ is a set of unbound parameters where $\{p_1, ..., p_k\} \subseteq P_{|B_{M_R}}$, (v) $\{p_{k+1}, ..., p_m\}$ is a set of bound parameters where $\{p_{k+1}, ..., p_m\} \subseteq P_{|B_{M_R}}$, (vi) $(c_{p_{k+1}}, ..., c_{p_m}) \in (B_{M_R})^{m-k}$, and (vii) $\{p_1, ..., p_k\} \cap \{p_{k+1}, ..., p_m\} = \emptyset$. Note that the form in Listing 88 is a "generic" form that is valid for any interpretation of the function name *filter*.

Listing 17: Function filter^{q_1}

$\mathit{filter}^{q_1} = \{c_1, ..., c_n, (p_1, ?\,), ..., (p_k, ?\,), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$

The evaluation and firing of o against q_1 yields the situation q_2, which has the function filter^{q_2} shown in Listing 104, where: (i) $n \geq 0$, (ii) $0 \leq k \leq m$, (iii) $\{c_1, ..., c_n\}$ is a set where $\{c_1, ..., c_n\} \subseteq B_{M_R}$, (iv) $\{p_1, ..., p_k\}$ is a set of unbound parameters where $\{p_1, ..., p_k\} \subseteq P_{|B_{M_R}}$, (v) $\{p_{k+1}, ..., p_m\}$ is a set of bound parameters where $\{p_{k+1}, ..., p_m\} \subseteq P_{|B_{M_R}}$, (vi) $(c_{p_{k+1}}, ..., c_{p_m}) \in (B_{M_R})^{m-k}$, and (vii) $\{p_1, ..., p_k\} \cap \{p_{k+1}, ..., p_m\} = \emptyset$. Note that (p, c) is guaranteed to be a member of filter^{q_2} since we excluded scenarios that led to empty update sets, shown in Listing 15, which means p is not a parameter in filter^{q_1}.

Listing 18: Function filter^{q_2}

$\mathit{filter}^{q_2} = \mathit{filter}^{q_1} \cup \{(p, c)\} = \{c_1, ..., c_n, (p_1, ?\,), ..., (p_k, ?\,), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m}), (p, c)\}$

A binding function π_{filter} (Definition 5.8) for filter^{q_1} has the form shown in Listing 105, where $c'_{p_i} \in dom_{\mathfrak{p}}(p_i)$ for each i such that $1 \leq i \leq k$. Note that we choose π_{filter} to bind all unbound parameters of filter^{q_1} since we require all binding tuples to be complete (refer to the convention after Example 5.8). Note, furthermore, that we choose π_{filter} to bind only unbound parameters of filter^{q_1} (see Section 7.2.1). Note that we do not need to consider the second property in Definition 5.8, related to dice levels and nodes, since π_{filter} does not bind these constituents.

Listing 19: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

We know from the definition of substitution (Definition 5.9), that the application of π_{filter} to the function filter^{q_1} of the situation q_1 replaces the binding of each parameter $p \in unbound(\mathit{filter}^{q_1})$ in filter^{q_1} with $\pi_{\mathit{filter}}(p)$, if $\pi_{\mathit{filter}}(p)$ is defined. The result of the described application is $\mathit{filter}^{q'_1}$ that has the form shown in Listing 106, where: (i) $n \geq 0$, (ii) $0 \leq k \leq m$, (iii) $\{c_1, ..., c_n\}$ is a set where $\{c_1, ..., c_n\} \subseteq B_{M_R}$, (iv) $\{p_1, ..., p_k\}$ is a set of bound parameters where $\{p_1, ..., p_k\} \subseteq P_{|B_{M_R}}$, (v) $(c'_{p_1}, ..., c'_{p_k}) \in (B_{M_R})^k$, (vi) $\{p_{k+1}, ..., p_m\}$ is a set of bound parameters where $\{p_{k+1}, ..., p_m\} \subseteq P_{|B_{M_R}}$, (vii) $(c_{p_{k+1}}, ..., c_{p_m}) \in (B_{M_R})^{m-k}$, and (viii) $\{p_1, ..., p_k\} \cap \{p_{k+1}, ..., p_m\} = \emptyset$.

Listing 20: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

The evaluation and firing of o' against q'_1 yields the situation q'_2, which has the function $\mathit{filter}^{q'_2}$ that has the form shown in Listing 108, where: (i) $n \geq 0$, (ii) $0 \leq k \leq m$, (iii) $\{c_1, ..., c_n\}$ is a set where $\{c_1, ..., c_n\} \subseteq B_{M_R}$, (iv) $\{p_1, ..., p_k\}$ is a set of bound parameters where $\{p_1, ..., p_k\} \subseteq P_{|B_{M_R}}$, (v) $(c'_{p_1}, ..., c'_{p_k}) \in (B_{M_R})^k$, (vi) $\{p_{k+1}, ..., p_m\}$ is a set of bound parameters where $\{p_{k+1}, ..., p_m\} \subseteq P_{|B_{M_R}}$, (vii) $(c_{p_{k+1}}, ..., c_{p_m}) \in (B_{M_R})^{m-k}$, and (viii) $\{p_1, ..., p_k\} \cap \{p_{k+1}, ..., p_m\} = \emptyset$. Note that (p, c') is guaranteed to be a member of $\mathit{filter}^{q'_2}$ since we excluded scenarios that led to empty update sets, shown in Listing 49, which means p is not a parameter in $\mathit{filter}^{q'_1}$.

Listing 21: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = \mathit{filter}^{q'_1} \cup \{(p, c')\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m}), (p, c')\}$$

Let b^1, shown in Listing 22, be a binding tuple (Definition 5.8) of the situation q_1, such that b^1 binds all unbound parameters of q_1 (refer to the convention after Example 5.8) and that b^1 binds only unbound parameters of q_1 (see Section 7.2.1). Also, b^1 must bind each parameter to a value of the parameter's domain (first property in Definition 5.8), and satisfy that the situation resulting from the application of b^1 to q_1 fulfills the second property in Definition 5.8, related to dice levels and nodes.

Listing 22: Binding tuple b^1

$$b^1 = ((\pi^{b^1}_{\mathit{measure}}, \pi^{b^1}_{\mathit{filter}}), ..., (\pi^{b^1}_{\mathit{gran}_i},\ \pi^{b^1}_{\mathit{level}_i},\ \pi^{b^1}_{\mathit{node}_i},\ \pi^{b^1}_{\mathit{selection}_i}), ...)$$

Since the operation $o = \textsc{AddResultPredicate}(p, c)$ targets only the function filter^{q_1} of the source situation q_1, it follows that q_2 may be different from q_1 only in the interpretation of the function name *filter*. Hence, the binding tuple b^2 shown in Listing 23 (that is identical to b^1 except for $\pi^{b^1}_{\mathit{filter}}$ that is replaced with $\pi^{b^2}_{\mathit{filter}}$) is a binding tuple of q_2 if $\pi^{b^2}_{\mathit{filter}}$ is a binding function for filter^{q_2}. In order to show that Definition 7.6 holds, we only concentrate on $\pi^{b^2}_{\mathit{filter}}$, i.e., a binding function for filter^{q_2}, since the binding tuple b^2 can be used for q_2 if $\pi^{b^2}_{\mathit{filter}}$ is a binding function for filter^{q_2}, which will satisfy Definition 7.6. Note that b^2 must also preserve the second property in Definition 5.8, related to dice levels and nodes, which is the case since: (i) b^1 preserves that property, (ii) the only difference between b^1 and b^2 is $\pi^{b^2}_{\mathit{filter}}$ which is not related to dice levels and nodes, and (iii) the only difference between q_1 and q_2 is the interpretation of the function name *filter*, which is also not related to dice levels and nodes.

Listing 23: Binding tuple b^2

$$b^2 = ((\pi^{b^1}_{\mathit{measure}}, \pi^{b^2}_{\mathit{filter}}), ..., (\pi^{b^1}_{\mathit{gran}_i},\ \pi^{b^1}_{\mathit{level}_i},\ \pi^{b^1}_{\mathit{node}_i},\ \pi^{b^1}_{\mathit{selection}_i}), ...)$$

Recall that the binding function $\pi^{b^2}_{\mathit{filter}}$ must bind all unbound parameters of filter^{q_2} (refer to the convention after Example 5.8) and must bind only unbound parameters of filter^{q_2} (see Section 7.2.1). Recall also that the binding function $\pi^{b^2}_{\mathit{filter}}$ must bind each parameter in the domain of $\pi^{b^2}_{\mathit{filter}}$ to a value that is a member of the domain of the parameter (refer to the conditions in Definition 5.8). Note that we do not need to consider the second property in Definition 5.8, related to dice levels and nodes, since $\pi^{b^2}_{\mathit{filter}}$ does not alter these constituents.

In order to prove semantics preservation (Definition 7.6), we need to show that there exists a binding function that when applied to filter^{q_2} (Listing 104) results in $\mathit{filter}^{q'_2}$ (Listing 108). Note that there exist two possible cases for filter^{q_2}, which are when (I) $c =?$ and when (II) $c \neq ?$. In the following, we state a binding function that satisfies semantics preservation, for both cases.

(I) $c =?$. In this case, we claim that the required binding function is $\pi_{\mathit{filter}} \cup \{p \mapsto c'\}$ (see Listing 105 for π_{filter}), which has the form shown in Listing 24. We prove that this claim is true in the following.

<u>First</u>, we need to show that $\pi_{\mathit{filter}} \cup \{p \mapsto c'\}$ is a binding function for filter^{q_2}. That is true because:

- $\pi_{\mathit{filter}} \cup \{p \mapsto c'\}$ binds all unbound parameters and only unbound parameters of filter^{q_2}, as can be seen in Listing 24.
- $\pi_{\mathit{filter}} \cup \{p \mapsto c'\}$ binds each parameter in the domain of the function $\pi_{\mathit{filter}} \cup \{p \mapsto c'\}$ to a value that is a member of the domain of the parameter, since:
  - For each i, where $1 \leq i \leq k$, it holds that $c'_{p_i} \in dom_{\mathtt{p}}(p_i)$ since we established that π_{filter} (Listing 105) is a binding function for filter^{q_1} (Listing 88).
  - For the remaining parameter p, the binding function $\pi_{\mathit{filter}} \cup \{p \mapsto c'\}$ binds p to c'. It holds that $c' \in dom_{\mathtt{p}}(p)$ since we excluded cases leading to empty update sets for the ground operation $o' = \textsc{AddResultPredicate}(p, c')$ (see Listing 49 for cases leading to empty update sets).

Listing 24: Binding function $\pi_{\mathit{filter}} \cup \{p \mapsto c'\}$

$$\pi_{\mathit{filter}} \cup \{p \mapsto c'\} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}, p \mapsto c'\}$$

Second, we need to show that the application of $\pi_{\mathit{filter}} \cup \{p \mapsto c'\}$ to filter^{q_2} results in $\mathit{filter}^{q_2'}$, which can be easily concluded by inspecting $\pi_{\mathit{filter}} \cup \{p \mapsto c'\}$ (Listing 24), filter^{q_2} (Listing 104), and $\mathit{filter}^{q_2'}$ (Listing 108).

(II) $c \neq ?$. In this case, we claim that the required binding function is π_{filter}. We prove that this claim is true in the following.

First, we need to show that π_{filter} is a binding function of filter^{q_2}. That is true since:

- π_{filter} binds all unbound parameters and only unbound parameters of filter^{q_2}, as can be seen in Listing 105 and Listing 104.
- π_{filter} binds each parameter in the domain of π_{filter} to a value that is a member of the domain of the parameter, since we established that π_{filter} is a binding for filter^{q_1}, which means that π_{filter} satisfies this condition (refer to the conditions in Definition 5.8).

Second, we need to show that the application of π_{filter} to filter^{q_2} results in $\mathit{filter}^{q_2'}$, which can be easily concluded by inspecting π_{filter} (Listing 105), filter^{q_2} (Listing 104), and $\mathit{filter}^{q_2'}$ (Listing 108).

□

3 Add Constant Result Predicate

Proposition 3. The operation type $t = (\textsc{AddResultPredicate}, (B_{M_R}), updSet_{\text{ADDRP}_2})$ (Table 6.2) is semantics-preserving.

Table 3: Definitions of items used in the proof

Item	Definition
c	An arbitrary value $c \in B_{M_R}$
o	A navigation operation $o = \textsc{AddResultPredicate}(c)$ of the type t, at the analysis graph level
o'	A ground operation $o' = o$ of the operation o, at the actual analysis level
$\pi = \emptyset$	A binding function of o such that $\beta^o(\pi) = o'$
q_1	An arbitrary analysis situation
b^1	An arbitrary binding tuple for q_1
q'_1	The situation resulting from the application of b^1 to q_1, i.e., $\beta^{q_1}(b^1) = q'_1$
q'_2	The situation resulting from the evaluation and firing of o' in q'_1, i.e., $\delta^{q'_1}(\epsilon^{q'_1}(o')) = q'_2$
q_2	The situation resulting from the evaluation and firing of o in q_1, i.e., $\delta^{q_1}(\epsilon^{q_1}(o)) = q_2$

Listing 25: Cases in which o results in an empty update set when applied to q_1

$$c \in nbConsts(\mathit{filter}^{q_1})$$

Listing 26: Cases in which o' results in an empty update set when applied to q'_1

$$c \in nbConsts(\mathit{filter}^{q'_1})$$

Listing 27: Function filter^{q_1}

$$\mathit{filter}^{q_1} = \{c_1, ..., c_n, (p_1, ?\,), ..., (p_k, ?\,), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 28: Function filter^{q_2}

$$\mathit{filter}^{q_2} = \mathit{filter}^{q_1} \cup \{c\} = \{c_1, ..., c_n, c, (p_1, ?\,), ..., (p_k, ?\,), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 29: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 30: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 31: Function $\mathit{filter}^{q_2'}$

$$\mathit{filter}^{q_2'} = \mathit{filter}^{q_1'} \cup \{c\} = \{c_1, ..., c_n, c, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 32: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

4 Remove Result Predicate

Proposition 4. The operation type $t = (\textsc{RemoveResultPredicate}, (P_{|B_{M_R}} \cup B_{M_R}), updSet_{\text{RMVRP}})$ (Table 6.2) is semantics-preserving.

Table 4: Definitions of items used in the proof

Item	Definition
v	An arbitrary value $v \in P_{\vert B_{M_R}} \cup B_{M_R}$
o	A navigation operation $o = \textsc{RemoveResultPredicate}(v)$ of the type t, at the analysis graph level
o'	A ground operation $o' = o$ of the operation o, at the actual analysis level
$\pi = \emptyset$	A binding function of o such that $\beta^{o}(\pi) = o'$
q_1	An arbitrary analysis situation
b^1	An arbitrary binding tuple for q_1
q_1'	The situation resulting from the application of b^1 to q_1, i.e., $\beta^{q_1}(b^1) = q_1'$
q_2'	The situation resulting from the evaluation and firing of o' in q_1', i.e., $\delta^{q_1'}(\epsilon^{q_1'}(o')) = q_2'$
q_2	The situation resulting from the evaluation and firing of o in q_1, i.e., $\delta^{q_1}(\epsilon^{q_1}(o)) = q_2$

Listing 33: Function filter^{q_1}

$$\mathit{filter}^{q_1} = \{c_1, ..., c_n, (p_1, ?), ..., (p_k, ?), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

4.1 v is a constant

Listing 34: Cases in which o results in an empty update set when applied to q_1

$$v \notin nbConsts(\mathit{filter}^{q_1})$$

Listing 35: Cases in which o' results in an empty update set when applied to q_1'

$$v \notin nbConsts(\mathit{filter}^{q_1'})$$

Let $v = c_n$.

Listing 36: Function filter^{q_2}

$$\mathit{filter}^{q_2} = \mathit{filter}^{q_1} \setminus \{c_n\} = \{c_1, ..., c_{n-1}, (p_1, ?), ..., (p_k, ?), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 37: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 38: Function $\mathit{filter}^{q_1'}$

$$\mathit{filter}^{q_1'} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 39: Function $\mathit{filter}^{q_2'}$

$$\mathit{filter}^{q_2'} = \mathit{filter}^{q_1'} \setminus \{c_n\} = \{c_1, ..., c_{n-1}, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 40: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

4.2 v is an unbound parameter

Listing 41: Cases in which o results in an empty update set when applied to q_1

$$v \notin paras(\mathit{filter}^{q_1})$$

Listing 42: Cases in which o' results in an empty update set when applied to q_1'

$$v \notin paras(\mathit{filter}^{q_1'})$$

Let $v = p_k$.

Listing 43: Function filter^{q_2}

$$\mathit{filter}^{q_2} = \mathit{filter}^{q_1} \setminus \{(p_k, ?)\} = \{c_1, ..., c_n, (p_1, ?), ..., (p_{k-1}, ?), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 44: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 45: Function $\mathit{filter}^{q_1'}$

$$\mathit{filter}^{q_1'} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 46: Function $\mathit{filter}^{q_2'}$

$$\mathit{filter}^{q_2'} = \mathit{filter}^{q_1'} \setminus \{(p_k, c'_{p_k})\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_{k-1}, c'_{p_{k-1}}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 47: Proof binding function

$$\pi_{\mathit{filter}} \setminus \{p_k \mapsto c'_{p_k}\} = \{p_1 \mapsto c'_{p_1}, ..., p_{k-1} \mapsto c'_{p_{k-1}}\}$$

4.3 v is a bound parameter

Let $v = p_m$.

Listing 48: Cases in which o results in an empty update set when applied to q_1

$$v \notin paras(\mathit{filter}^{q_1})$$

Listing 49: Cases in which o' results in an empty update set when applied to q'_1

$$v \notin paras(\mathit{filter}^{q'_1})$$

Listing 50: Function filter^{q_2}

$$\mathit{filter}^{q_2} = \mathit{filter}^{q_1} \setminus \{(p_m, c_{p_m})\} = \{c_1, ..., c_n, (p_1, ?\,), ..., (p_k, ?\,), (p_{k+1}, c_{p_{k+1}}), ..., (p_{m-1}, c_{p_{m-1}})\}$$

Listing 51: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 52: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 53: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = \mathit{filter}^{q'_1} \setminus \{(p_m, c_{p_m})\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_{m-1}, c_{p_{m-1}})\}$$

Listing 54: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

5 Replace Result-Predicate Pair

Proposition 5. The operation type $t = (\textsc{ReplaceResultPredicate}, (P_{|B_{M_R}}, P_{|B_{M_R}}, B_{M_R} \ \cup \ U), \mathit{updSet}_{\text{RPLRP}_1})$ (Table 6.2) is semantics-preserving.

Table 5: Definitions of items used in the proof

Item	Definition
p_a	An arbitrary parameter $p_a \in P_{\|B_{M_R}}$
p_b	An arbitrary parameter $p_b \in P_{\|B_{M_R}}$
c_b	An arbitrary value $c_b \in B_{M_R} \cup U$
c'_b	An arbitrary value $c'_b \in B_{M_R}$
o	A navigation operation $o = \textsc{ReplaceResultPredicate}(p_a, p_b, c_b)$ of the type t, at the analysis graph level
o'	A ground operation $o' = \textsc{ReplaceResultPredicate}(p_a, p_b, c'_b)$ of the operation o, at the actual analysis level
π	A binding function of o such that $\beta^o(\pi) = o'$
q_1	An arbitrary analysis situation
b^1	An arbitrary binding tuple for q_1
q'_1	The situation resulting from the application of b^1 to q_1, i.e., $\beta^{q_1}(b^1) = q'_1$
q'_2	The situation resulting from the evaluation and firing of o' in q'_1, i.e., $\delta^{q'_1}(\epsilon^{q'_1}(o')) = q'_2$
q_2	The situation resulting from the evaluation and firing of o in q_1, i.e., $\delta^{q_1}(\epsilon^{q_1}(o)) = q_2$

Listing 55: Cases in which o results in an empty update set when applied to q_1

$p_a \notin paras(\mathit{filter}^{q_1})$
$p_b \in paras(\mathit{filter}^{q_1})$
$c_b \notin dom_{\mathfrak{p}}(p_b) \cup U$

Listing 56: Cases in which o' results in an empty update set when applied to q_1

$p_a \notin paras(\mathit{filter}^{q'_1})$
$p_b \in paras(\mathit{filter}^{q'_1})$
$c'_b \notin dom_{\mathfrak{p}}(p_2) \cup U$

Listing 57: Function filter^{q_1}

$\mathit{filter}^{q_1} = \{c_1, ..., c_n, (p_1, ?), ..., (p_k, ?), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$

5.1 p_a is unbound and $c_b = ?$

Let $p_a = p_k$.

Listing 58: Function filter^{q_2}

$$\mathit{filter}^{q_2} = (\mathit{filter}^{q_1} \setminus \{(p_k, ?)\}) \cup \{(p_b, ?)\} = \{c_1, ..., c_n, (p_1, ?), ..., (p_{k-1}, ?), \ (p_b, ?), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 59: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 60: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 61: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = (\mathit{filter}^{q'_1} \setminus \{(p_k, c'_{p_k})\}) \cup \{(p_b, c'_b)\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_{k-1}, c'_{p_{k-1}}), (p_b, c'_b), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 62: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_{k-1} \mapsto c'_{p_{k-1}}, p_b \mapsto c'_b\}$$

5.2 p_a is bound and $c_b = ?$

Let $p_a = p_m$.

Listing 63: Function filter^{q_2}

$$\mathit{filter}^{q_2} = (\mathit{filter}^{q_1} \setminus \{(p_m, c_{p_m})\}) \cup \{(p_b, ?)\} = \{c_1, ..., c_n, (p_1, ?), ..., (p_k, ?), \ (p_b, ?), (p_{k+1}, c_{p_{k+1}}), ..., (p_{m-1}, c_{p_{m-1}})\}$$

Listing 64: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 65: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 66: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = (\mathit{filter}^{q'_1} \setminus \{(p_m, c_{p_m})\}) \cup \{(p_b, c'_b)\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_b, c'_b), (p_{k+1}, c_{p_{k+1}}), ..., (p_{m-1}, c_{p_{m-1}})\}$$

Listing 67: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p \mapsto c'_{p_k}, p_b \mapsto c'_b\}$$

5.3 p_a is unbound and c_b is a constant

Let $p_a = p_k$.

Listing 68: Function filter^{q_2}

$$\mathit{filter}^{q_2} = (\mathit{filter}^{q_1} \setminus \{(p_k, ?\,)\}) \cup \{(p_b, c_b)\} = \{c_1, ..., c_n, (p_1, ?\,), ..., (p_{k-1}, ?\,),\ \ (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m}), (p_b, c_b)\}$$

Listing 69: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 70: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 71: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = (\mathit{filter}^{q'_1} \setminus \{(p_k, c'_{p_k})\}) \cup \{(p_b, c_b)\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_{k-1}, c'_{p_{k-1}})), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m}), (p_b, c_b)\}$$

Listing 72: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_{k-1} \mapsto c'_{p_{k-1}}\}$$

5.4 p_a is bound and c_b is a constant

Let $p_a = p_m$.

Listing 73: Function filter^{q_2}

$$\mathit{filter}^{q_2} = (\mathit{filter}^{q_1} \setminus \{(p_m, c_{p_m})\}) \cup \{(p_b, c_b)\} = \{c_1, ..., c_n, (p_1, ?\,), ..., (p_k, ?\,),\ \ (p_{k+1}, c_{p_{k+1}}), ..., (p_{m-1}, c_{p_{m-1}}),\ \ (p_b, c_b))\}$$

Listing 74: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 75: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 76: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = (\mathit{filter}^{q'_1} \setminus \{(p_m, c_{p_m})\}) \cup \{(p_b, c_b)\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_{m-1}, c_{p_{m-1}}), (p_b, c_b)\}$$

Listing 77: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p \mapsto c'_{p_k}\}$$

6 Replace Constant Result Predicate

Proposition 6. The operation type $t = (\textsc{ReplaceResultPredicate}, (B_{M_R}, B_{M_R}),\ updSet_{\mathrm{RplRP}_2})$ (Table 6.2) is semantics-preserving.

Table 6: Definitions of items used in the proof

Item	Definition
c_a	An arbitrary value $c_a \in B_{M_R}$
c_b	An arbitrary value $c_b \in B_{M_R}$
o	A navigation operation $o = \textsc{ReplaceResultPredicate}(c_a, c_b)$ of the type t, at the analysis graph level
o'	A ground operation $o' = o$ of the operation o, at the actual analysis level
$\pi = \emptyset$	A binding function of o such that $\beta^o(\pi) = o'$
q_1	An arbitrary analysis situation
b^1	An arbitrary binding tuple for q_1
q_1'	The situation resulting from the application of b^1 to q_1, i.e., $\beta^{q_1}(b^1) = q_1'$
q_2'	The situation resulting from the evaluation and firing of o' in q_1', i.e., $\delta^{q_1'}(\epsilon^{q_1'}(o')) = q_2'$
q_2	The situation resulting from the evaluation and firing of o in q_1, i.e., $\delta^{q_1}(\epsilon^{q_1}(o)) = q_2$

Listing 78: Cases in which o results in an empty update set when applied to q_1

$c_a \notin nbConsts(\mathit{filter}^{q_1})$
$c_b \in nbConsts(\mathit{filter}^{q_1})$

Listing 79: Cases in which o' results in an empty update set when applied to q_1

$c_a \notin nbConsts(\mathit{filter}^{q_1'})$
$c_b \in nbConsts(\mathit{filter}^{q_1'})$

Listing 80: Function filter^{q_1}

$\mathit{filter}^{q_1} = \{c_1, ..., c_n, (p_1, ?\,), ..., (p_k, ?\,), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$

Let $c_a = c_n$.

Listing 81: Function filter^{q_2}

$\mathit{filter}^{q_2} = (\mathit{filter}^{q_1} \setminus \{c_a\}) \cup \{c_b\} = \{c_1, ..., c_{n-1}, c_b, (p_1, ?\,), ..., (p_k, ?\,),\ (p_{k+1}, c_{p_{k+1}}), ...,\ (p_m, c_{p_m})\}$

Listing 82: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 83: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 84: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = (\mathit{filter}^{q'_1} \setminus \{c_n\}) \cup \{c_b)\} = \{c_1, ..., c_{n-1}, c_b, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_b, c'_b), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 85: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

7 Rebind Result Predicate

Proposition 7. The operation type $t = (\textsc{RebindResultPredicate}, (P_{|B_{M_R}}, B_{M_R} \cup U),\ updSet_{\text{RBDRP}})$ (Table 6.2) is semantics-preserving.

Table 7: Definitions of items used in the proof

Item	Definition
p	An arbitrary parameter $p \in P_{\vert B_{M_R}}$
c	An arbitrary value $c \in B_{M_R} \cup U$
c'	An arbitrary value $c' \in B_{M_R}$
o	A navigation operation $o = \textsc{RebindResultPredicate}(p, c)$ of the type t, at the analysis graph level
o'	A ground operation $o' = \textsc{RebindResultPredicate}(p, c')$ of the operation o, at the actual analysis level
π	A binding function of o such that $\beta^o(\pi) = o'$
q_1	An arbitrary analysis situation
b^1	An arbitrary binding tuple for q_1
q'_1	The situation resulting from the application of b^1 to q_1, i.e., $\beta^{q_1}(b^1) = q'_1$
q'_2	The situation resulting from the evaluation and firing of o' in q'_1, i.e., $\delta^{q'_1}(\epsilon^{q'_1}(o')) = q'_2$
q_2	The situation resulting from the evaluation and firing of o in q_1, i.e., $\delta^{q_1}(\epsilon^{q_1}(o)) = q_2$

Listing 86: Cases in which o results in an empty update set when applied to q_1

$p \notin paras(\mathit{filter}^{q_1})$
$c \notin dom_{\mathfrak{p}}(p) \cup U$

Listing 87: Cases in which o' results in an empty update set when applied to q_1

$p \notin paras(\mathit{filter}^{q'_1})$
$c' \notin dom_{\mathfrak{p}}(p) \cup U$

Listing 88: Function filter^{q_1}

$\mathit{filter}^{q_1} = \{c_1, ..., c_n, (p_1, ?\,), ..., (p_k, ?\,), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$

7.1 p is unbound and $c = ?$

Let $p = p_k$.

Listing 89: Function filter^{q_2}

$$\mathit{filter}^{q_2} = (\mathit{filter}^{q_1} \setminus \{(p_k, ?)\}) \cup \{(p_k, ?)\} = \{c_1, ..., c_n, (p_1, ?), ..., (p_k, ?), \ (p_{k+1}, c_{p_{k+1}}), ..., \ (p_m, c_{p_m})\}$$

Listing 90: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 91: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 92: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = (\mathit{filter}^{q'_1} \setminus \{(p_k, c'_{p_k})\}) \cup \{(p_k, c')\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'),(p_{k+1}, c_{p_{k+1}}),..., (p_m, c_{p_m})\}$$

Listing 93: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'\}$$

7.2 p is bound and $c = ?$

Let $p = p_m$.

Listing 94: Function filter^{q_2}

$$\mathit{filter}^{q_2} = (\mathit{filter}^{q_1} \setminus \{(p_m, c_{p_m})\}) \cup \{(p_m, ?)\} = \{c_1, ..., c_n, (p_1, ?), ..., (p_k, ?), \ (p_{k+1}, c_{p_{k+1}}), ..., \ (p_m, ?)\}$$

Listing 95: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 96: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 97: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = (\mathit{filter}^{q'_1} \setminus \{(p_m, c_{p_m})\}) \cup \{(p_m, c')\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), \ (p_{k+1}, c_{p_{k+1}}), \\ ..., (p_m, c')\}$$

Listing 98: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}, p_m \mapsto c'\}$$

7.3 p is unbound and c is a constant

Let $p = p_k$.

Listing 99: Function filter^{q_2}

$$\mathit{filter}^{q_2} = (\mathit{filter}^{q_1} \setminus \{(p_k, ?)\}) \cup \{(p_k, c)\} = \{c_1, ..., c_n, (p_1, ?), ..., (p_k, c),\ (p_{k+1}, c_{p_{k+1}}), ...,\ (p_m, c_{p_m})\}$$

Listing 100: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 101: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 102: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = (\mathit{filter}^{q'_1} \setminus \{(p_k, c'_{p_k})\}) \cup \{(p_k, c)\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c), (p_{k+1}, c_{p_{k+1}}),\ ..., (p_m, c_{p_m})\}$$

Listing 103: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_{k-1} \mapsto c'_{p_{k-1}}\}$$

7.4 p is bound and c is a constant

Let $p = p_m$.

Listing 104: Function filter^{q_2}

$$\mathit{filter}^{q_2} = (\mathit{filter}^{q_1} \setminus \{(p_m, c_{p_m})\}) \cup \{(p_m, c)\} = \{c_1, ..., c_n, (p_1, ?), ..., (p_k, ?),\ (p_{k+1}, c_{p_{k+1}}), ...,\ (p_m, c)\}$$

Listing 105: Binding function π_{filter}

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

Listing 106: Function $\mathit{filter}^{q'_1}$

$$\mathit{filter}^{q'_1} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c_{p_m})\}$$

Listing 107: Function $\mathit{filter}^{q'_2}$

$$\mathit{filter}^{q'_2} = (\mathit{filter}^{q'_1} \setminus \{(p_m, c_{p_m})\}) \cup \{(p_m, c)\} = \{c_1, ..., c_n, (p_1, c'_{p_1}), ..., (p_k, c'_{p_k}), (p_{k+1}, c_{p_{k+1}}), ..., (p_m, c)\}$$

Listing 108: Proof binding function

$$\pi_{\mathit{filter}} = \{p_1 \mapsto c'_{p_1}, ..., p_k \mapsto c'_{p_k}\}$$

8 Roll Up (Drill Down)

$o = \textsc{RollUp}_i(h)$ and $o' = \textsc{RollUp}_i(h)$.

8.1 When $\mathit{gran}_i^{q_1} = l \in C$

Listing 109: Cases in which o results in an empty update set when applied to q_1

$l = top_i$ **or** $(h, l) \notin HL$

Listing 110: Cases in which o' results in an empty update set when applied to q'_1

$l = top_i$ **or** $(h, l) \notin HL$

Table 8

	Source Situation 1	Target Situation 2
Schema q	$\mathit{gran}_i^{q_1} = l$	$\mathit{gran}_i^{q_2} = nextLevel_i(h, l)$
Trace q'	$\mathit{gran}_i^{q'_1} = l$	$\mathit{gran}_i^{q'_2} = nextLevel_i(h, l)$

8.2 When $\mathit{gran}_i^{q_1} = (p, l) \in Pr$ where $l \in C$

Listing 111: Cases in which o results in an empty update set when applied to q_1

$l = top_i$ **or** $(h, l) \notin HL$ **or** $nextLevel_i(h, l) \notin dom_{\mathtt{p}}(p)$

Listing 112: Cases in which o' results in an empty update set when applied to q'_1

$l = top_i$ **or** $(h, l) \notin HL$ **or** $nextLevel_i(h, l) \notin dom_{\mathtt{p}}(p)$

Table 9

	Source Situation 1	Target Situation 2
Schema q	$\mathit{gran}_i^{q_1} = (p, l)$	$\mathit{gran}_i^{q_2} = (p, nextLevel_i(h, l))$
Trace q'	$\mathit{gran}_i^{q'_1} = (p, l)$	$\mathit{gran}_i^{q'_2} = (p, nextLevel_i(h, l))$

8.3 When $\mathit{gran}_i^{q_1} = (p, ?) \in Pr$

Let p be bound to a value l' in q'_1.

Listing 113: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 114: Cases in which o' results in an empty update set when applied to q'_1

```
l' = top_i  or  (h, l') ∉ HL  or  nextLevel_i(h, l') ∉ dom_p(p)
```

Table 10

	Source Situation 1	Target Situation 2
Schema q	$\mathsf{gran}_i^{q_1} = (p, ?)$	$\mathsf{gran}_i^{q_2} = (p, ?)$
Trace q'	$\mathsf{gran}_i^{q'_1} = (p, l')$	$\mathsf{gran}_i^{q'_2} = (p, nextLevel_i(h, l'))$

Listing 115: Proof binding function

```
π_gran = {p ↦ nextLevel_i(h, l')}
```

9 Roll Up To (Drill Down To)

$o = \textsc{RollUpTo}_i(l_r)$ and $o' = \textsc{RollUpTo}_i(l_r)$.

9.1 When $\textit{gran}_i^{q_1} = l_d \in C$

Listing 116: Cases in which o results in an empty update set when applied to q_1

```
(l_d, l_r) ∉ LL_i^+ or l_d = l_r
```

Listing 117: Cases in which o' results in an empty update set when applied to q_1

```
(l_d, l_r) ∉ LL_i^+ or l_d = l_r
```

Table 11

	Source Situation 1	Target Situation 2
Schema q	$\textit{gran}_i^{q_1} = l_d$	$\textit{gran}_i^{q_2} = l_r$
Trace q'	$\textit{gran}_i^{q'_1} = l_d$	$\textit{gran}_i^{q'_2} = l_r$

9.2 When $\textit{gran}_i^{q_1} = (p, l_d) \in Pr$ where $l_d \in C$

Listing 118: Cases in which o results in an empty update set when applied to q_1

```
(l_d, l_r) ∉ LL_i^+ or l_d = l_r or l_r ∉ dom_p(p)
```

Listing 119: Cases in which o' results in an empty update set when applied to q'_1

```
(l_d, l_r) ∉ LL_i^+ or l_d = l_r or l_r ∉ dom_p(p)
```

Table 12

	Source Situation 1	Target Situation 2
Schema q	$\textit{gran}_i^{q_1} = (p, l_d)$	$\textit{gran}_i^{q_2} = (p, l_r)$
Trace q'	$\textit{gran}_i^{q'_1} = (p, l_d)$	$\textit{gran}_i^{q'_2} = (p, l_r)$

9.3 When $\textit{gran}_i^{q_1} = (p, ?\,) \in Pr$

Never fires at SWAG level, which allows to exclude this case.

10 Set Granularity

r and d in level names denote new and old, respectively (not up and down as in previous section).

$o = \textsc{SetGranularity}_i(l_r)$ and $o' = \textsc{SetGranularity}_i(l_r)$.

10.1 When $\textit{gran}_i^{q_1} = l_d \in C$

Listing 120: Cases in which o results in an empty update set when applied to q_1

$l_d = l_r$

Listing 121: Cases in which o' results in an empty update set when applied to q_1

$l_d = l_r$

Table 13

	Source Situation 1	Target Situation 2
Schema q	$\textit{gran}_i^{q_1} = l_d$	$\textit{gran}_i^{q_2} = l_r$
Trace q'	$\textit{gran}_i^{q'_1} = l_d$	$\textit{gran}_i^{q'_2} = l_r$

10.2 When $\textit{gran}_i^{q_1} = (p, l_d) \in Pr$ where $l_d \in C$

Listing 122: Cases in which o results in an empty update set when applied to q_1

$l_r \notin dom_{\mathtt{p}}(p)$

Listing 123: Cases in which o' results in an empty update set when applied to q'_1

$l_r \notin dom_{\mathtt{p}}(p)$

Table 14

	Source Situation 1	Target Situation 2
Schema q	$\textit{gran}_i^{q_1} = (p, l_d)$	$\textit{gran}_i^{q_2} = (p, l_r)$
Trace q'	$\textit{gran}_i^{q'_1} = (p, l_d)$	$\textit{gran}_i^{q'_2} = (p, l_r)$

10.3 When $\textit{gran}_i^{q_1} = (p, ?) \in Pr$

Let q'_1 bind p to l'_d.

Listing 124: Cases in which o results in an empty update set when applied to q_1

$l_r \notin dom_{\mathtt{p}}(p)$

Listing 125: Cases in which o' results in an empty update set when applied to q'_1

$$l_r \notin dom_{\mathtt{p}}(p)$$

Table 15

	Source Situation 1	Target Situation 2
Schema q	$\mathit{gran}_i^{q_1} = (p, ?\,)$	$\mathit{gran}_i^{q_2} = (p, l_r)$
Trace q'	$\mathit{gran}_i^{q'_1} = (p, l'_d)$	$\mathit{gran}_i^{q'_2} = (p, l_r)$

Listing 126: Proof binding function

$$\pi_{\mathit{gran}} = \emptyset$$

11 Reset Granularity (1)

r and d in level names denote new and old, respectively (not up and down as in previous section).

$o = \textsc{ResetGranularity}_i(l_r)$ and $o' = \textsc{ResetGranularity}_i(l_r)$.

11.1 When $\textit{gran}_i^{q_1} = l_d \in C$

Listing 127: Cases in which o results in an empty update set when applied to q_1

$l_d = l_r$

Listing 128: Cases in which o' results in an empty update set when applied to q'_1

$l_d = l_r$

Table 16

	Source Situation 1	Target Situation 2
Schema q	$\textit{gran}_i^{q_1} = l_d$	$\textit{gran}_i^{q_2} = l_r$
Trace q'	$\textit{gran}_i^{q'_1} = l_d$	$\textit{gran}_i^{q'_2} = l_r$

11.2 When $\textit{gran}_i^{q_1} = (p, l_d) \in Pr$ where $l_d \in C$

Listing 129: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 130: Cases in which o' results in an empty update set when applied to q'_1

```
None
```

Table 17

	Source Situation 1	Target Situation 2
Schema q	$\textit{gran}_i^{q_1} = (p, l_d)$	$\textit{gran}_i^{q_2} = l_r$
Trace q'	$\textit{gran}_i^{q'_1} = (p, l_d)$	$\textit{gran}_i^{q'_2} = l_r$

11.3 When $\textit{gran}_i^{q_1} = (p, ?) \in Pr$

Let q'_1 bind p to l'_d.

Listing 131: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 132: Cases in which o' results in an empty update set when applied to q_1'

```
None
```

Table 18

	Source Situation 1	Target Situation 2
Schema q	$\mathit{gran}_i^{q_1} = (p, ?\,)$	$\mathit{gran}_i^{q_2} = l_r$
Trace q'	$\mathit{gran}_i^{q_1'} = (p, l_d')$	$\mathit{gran}_i^{q_2'} = l_r$

12 Reset Granularity (2)

r and d in level names denote new and old, respectively (not up and down as in previous section).

$o = \textsc{ResetGranularity}_i(p_r, l_r)$ and $o' = \textsc{ResetGranularity}_i(p_r, l_r)$.

12.1 When $l_r \in C$

12.1.1 When $\textit{gran}_i^{q_1} = l_d \in C$

Listing 133: Cases in which o results in an empty update set when applied to q_1

$$l_r \notin dom_{\mathfrak{p}}(p_r) \cup U$$

Listing 134: Cases in which o' results in an empty update set when applied to q_1

$$l_r \notin dom_{\mathfrak{p}}(p_r) \cup U$$

Table 19

	Source Situation 1	Target Situation 2
Schema q	$\textit{gran}_i^{q_1} = l_d$	$\textit{gran}_i^{q_2} = (p_r, l_r)$
Trace q'	$\textit{gran}_i^{q_1'} = l_d$	$\textit{gran}_i^{q_2'} = (p_r, l_r)$

12.1.2 When $\textit{gran}_i^{q_1} = (p, l_d) \in Pr$ where $l_d \in C$

Listing 135: Cases in which o results in an empty update set when applied to q_1

$$l_r \notin dom_{\mathfrak{p}}(p_r) \cup U$$

Listing 136: Cases in which o' results in an empty update set when applied to q_1'

$$l_r \notin dom_{\mathfrak{p}}(p_r) \cup U$$

Table 20

	Source Situation 1	Target Situation 2
Schema q	$\textit{gran}_i^{q_1} = (p, l_d)$	$\textit{gran}_i^{q_2} = (p_r, l_r)$
Trace q'	$\textit{gran}_i^{q_1'} = (p, l_d)$	$\textit{gran}_i^{q_2'} = (p_r, l_r)$

12.1.3 When $\textit{gran}_i^{q_1} = (p, ?) \in Pr$

Let q_1' bind p to l_d'.

Listing 137: Cases in which o results in an empty update set when applied to q_1

$$l_r \notin dom_{\mathfrak{p}}(p_r) \cup U$$

Listing 138: Cases in which o' results in an empty update set when applied to q_1'

$l_r \notin dom_{\mathfrak{p}}(p_r) \cup U$

Table 21

	Source Situation 1	Target Situation 2
Schema q	$\mathit{gran}_i^{q_1} = (p, ?)$	$\mathit{gran}_i^{q_2} = (p_r, l_r)$
Trace q'	$\mathit{gran}_i^{q_1'} = (p, l_d')$	$\mathit{gran}_i^{q_2'} = (p_r, l_r)$

12.2 When $l_r =?$

12.2.1 When $\mathit{gran}_i^{q_1} = l_d \in C$

Let π_o bind p_r to l_r'.

Listing 139: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 140: Cases in which o' results in an empty update set when applied to q_1'

$l_r' \notin dom_{\mathfrak{p}}(p_r)$

Table 22

	Source Situation 1	Target Situation 2
Schema q	$\mathit{gran}_i^{q_1} = l_d$	$\mathit{gran}_i^{q_2} = (p_r, ?)$
Trace q'	$\mathit{gran}_i^{q_1'} = l_d$	$\mathit{gran}_i^{q_2'} = (p_r, l_r')$

Listing 141: Proof binding function

$\pi\mathit{gran} = \{p_r \mapsto l_r'\}$

12.2.2 When $\mathit{gran}_i^{q_1} = (p, l_d) \in Pr$ where $l_d \in C$

Listing 142: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 143: Cases in which o' results in an empty update set when applied to q_1'

$l_r' \notin dom_{\mathfrak{p}}(p_r)$

Listing 144: Proof binding function

$\pi\mathit{gran} = \{p_r \mapsto l_r'\}$

Table 23

	Source Situation 1	Target Situation 2
Schema q	$\mathit{gran}_i^{q_1} = (p, l_d)$	$\mathit{gran}_i^{q_2} = (p_r, ?)$
Trace q'	$\mathit{gran}_i^{q'_1} = (p, l_d)$	$\mathit{gran}_i^{q'_2} = (p_r, l'_r)$

12.2.3 When $\mathit{gran}_i^{q_1} = (p, ?) \in Pr$

Let q'_1 bind p to l'_d.

Listing 145: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 146: Cases in which o' results in an empty update set when applied to q'_1

$$l'_r \notin dom_{\mathtt{p}}(p_r)$$

Table 24

	Source Situation 1	Target Situation 2
Schema q	$\mathit{gran}_i^{q_1} = (p, ?)$	$\mathit{gran}_i^{q_2} = (p_r, ?)$
Trace q'	$\mathit{gran}_i^{q'_1} = (p, l'_d)$	$\mathit{gran}_i^{q'_2} = (p_r, l'_r)$

Listing 147: Proof binding function

$$\pi_{\mathit{gran}} = \{p_r \mapsto l'_r\}$$

13 Move To Next Node (Move To Previous Node)

$o = \textsc{MoveToNextNode}_i$ and $o' = \textsc{MoveToNextNode}_i$.

13.1 When $\mathit{level}_i^{q_1} = l \in C$

13.1.1 When $\mathit{node}_i^{q_1} = n \in C$

Listing 148: Cases in which o results in an empty update set when applied to q_1

$n = last_l$ **or** $nextMember(l, n) \notin \lambda(l)$

Listing 149: Cases in which o' results in an empty update set when applied to q_1'

$n = last_l$ **or** $nextMember(l, n) \notin \lambda(l)$

Table 25

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = n$	$\mathit{node}_i^{q_2} = nextMember_i(l, n)$
Trace q'	$\mathit{node}_i^{q_1'} = n$	$\mathit{node}_i^{q_2'} = nextMember_i(l, n)$

13.1.2 When $\mathit{node}_i^{q_1} = (p, n) \in Pr$ where $n \in C$

Listing 150: Cases in which o results in an empty update set when applied to q_1

$n = last_l$ **or** $nextMember_i(n) \notin dom_p(p)$ **or** $nextMember_i(n) \notin \lambda(l)$

Listing 151: Cases in which o' results in an empty update set when applied to q_1'

$n = last_l$ **or** $nextMember_i(n) \notin dom_p(p)$ **or** $nextMember_i(n) \notin \lambda(l)$

Table 26

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p, n)$	$\mathit{node}_i^{q_2} = (p, nextMember_i(n))$
Trace q'	$\mathit{node}_i^{q_1'} = (p, n)$	$\mathit{node}_i^{q_2'} = (p, nextMember_i(n))$

13.1.3 When $\mathit{node}_i^{q_1} = (p, ?\,) \in Pr$

Let p be bound to a value n' in q_1'.

Listing 152: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 153: Cases in which o' results in an empty update set when applied to q_1'

$n' = last_l$ **or** $nextMember_i(n') \notin dom_p(p)$ **or** $nextMember_i(n') \notin \lambda(l)$

Table 27

	Source Situation 1	Target Situation 2
Schema q	$node_i^{q_1} = (p, ?)$	$node_i^{q_2} = (p, ?)$
Trace q'	$node_i^{q_1'} = (p, n')$	$node_i^{q_2'} = (p, nextMember_i(n'))$

Listing 154: Proof binding function

$\pi_{node} = \{p \mapsto nextMember_i(n')\}$

13.2 When $level_i^{q_1} = (p_l, l) \in Pr$ and $l \in C$

13.2.1 When $node_i^{q_1} = n \in C$

Listing 155: Cases in which o results in an empty update set when applied to q_1

$n = last_l$ **or** $nextMember(l, n) \notin \lambda(l)$

Listing 156: Cases in which o' results in an empty update set when applied to q_1'

$n = last_l$ **or** $nextMember(l, n) \notin \lambda(l)$

Table 28

	Source Situation 1	Target Situation 2
Schema q	$node_i^{q_1} = n$	$node_i^{q_2} = nextMember_i(n)$
Trace q'	$node_i^{q_1'} = n$	$node_i^{q_2'} = nextMember_i(n)$

13.2.2 When $node_i^{q_1} = (p_n, n) \in Pr$ where $n \in C$

Listing 157: Cases in which o results in an empty update set when applied to q_1

$n = last_l$ **or** $nextMember_i(n) \notin dom_p(p_n)$ **or** $nextMember_i(n) \notin \lambda(l)$

Listing 158: Cases in which o' results in an empty update set when applied to q_1'

$n = last_l$ **or** $nextMember_i(n) \notin dom_p(p_n)$ **or** $nextMember_i(n) \notin \lambda(l)$

Table 29

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p_n, n)$	$\mathit{node}_i^{q_2} = (p_n, nextMember_i(n))$
Trace q'	$\mathit{node}_i^{q'_1} = (p_n, n)$	$\mathit{node}_i^{q'_2} = (p_n, nextMember_i(n))$

13.2.3 When $\mathit{node}_i^{q_1} = (p_n, ?\,) \in Pr$

Let p_n be bound to a value n' in q'_1.

Listing 159: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 160: Cases in which o' results in an empty update set when applied to q'_1

$n' = last_l$ **or** $nextMember_i(n') \notin dom_p(p_n)$ **or** $nextMember_i(n') \notin \lambda(l)$

Table 30

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p_n, ?\,)$	$\mathit{node}_i^{q_2} = (p_n, ?\,)$
Trace q'	$\mathit{node}_i^{q'_1} = (p_n, n')$	$\mathit{node}_i^{q'_2} = (p_n, nextMember_i(n'))$

Listing 161: Proof binding function

$\pi_{\mathit{node}} = \{p_n \mapsto nextMember_i(n')\}$

13.3 When $\mathit{level}_i^{q_1} = (p_l, ?\,) \in Pr$

13.3.1 When $\mathit{node}_i^{q_1} = n \in C$

Not possible to have unbound level and a constant node.

13.3.2 When $\mathit{node}_i^{q_1} = (p_n, n) \in Pr$ where $n \in C$

Not possible to have unbound level and a constant node.

13.3.3 When $\mathit{node}_i^{q_1} = (p_n, ?\,) \in Pr$

Let p_l be bound to l' in q'_1.

Let p_n be bound to a value n' in q'_1.

Listing 162: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 163: Cases in which o' results in an empty update set when applied to q_1'

$n' = last_l$ **or** $nextMember_i(n') \notin dom_p(p_n)$ **or** $nextMember_i(n') \notin \lambda(l')$

Table 31

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = (p_l, ?)$, $\mathit{node}_i^{q_1} = (p_n, ?)$	$\mathit{level}_i^{q_2} = (p_l, ?)$, $\mathit{node}_i^{q_2} = (p_n, ?)$
Trace q'	$\mathit{level}_i^{q_1} = (p_l, l')$, $\mathit{node}_i^{q_1'} = (p_n, n')$	$\mathit{level}_i^{q_2} = (p_l, l')$ $\mathit{node}_i^{q_2'} = (p, nextMember_i(n'))$

Listing 164: Proof binding function

$\pi_{\mathit{node}} = \{p_l \mapsto l', p_n \mapsto nextMember_i(n')\}$

14 Change Node To

$o = \textsc{ChangeNodeTo}_i(n_r)$ and $o' = \textsc{ChangeNodeTo}_i(n_r)$.

14.1 When $\textit{level}_i^{q_1} = l \in C$

14.1.1 When $\textit{node}_i^{q_1} = n_d \in C$

Listing 165: Cases in which o results in an empty update set when applied to q_1

```
n_d = n_r or n_r ∉ λ(l)
```

Listing 166: Cases in which o' results in an empty update set when applied to q'_1

```
n_d = n_r or n_r ∉ λ(l)
```

Table 32

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = n_d$	$\textit{node}_i^{q_2} = n_r$
Trace q'	$\textit{node}_i^{q'_1} = n_d$	$\textit{node}_i^{q'_2} = n_r$

14.1.2 When $\textit{node}_i^{q_1} = (p, n_d) \in Pr$ where $n_d \in C$

Listing 167: Cases in which o results in an empty update set when applied to q_1

```
n_r ∉ λ(l) or n_r ∉ dom_p(p)
```

Listing 168: Cases in which o' results in an empty update set when applied to q'_1

```
n_r ∉ λ(l) or n_r ∉ dom_p(p)
```

Table 33

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = (p, n_d)$	$\textit{node}_i^{q_2} = (p, n_r)$
Trace q'	$\textit{node}_i^{q'_1} = (p, n_d)$	$\textit{node}_i^{q'_2} = (p, n_r)$

14.1.3 When $\textit{node}_i^{q_1} = (p, ?) \in Pr$

Let p be bound to a value n'_d in q'_1.

Listing 169: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 170: Cases in which o' results in an empty update set when applied to q'_1

```
n_r ∉ λ(l) or n_r ∉ dom_p(p)
```

Table 34

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p, ?\,)$	$\mathit{node}_i^{q_2} = (p, n_r)$
Trace q'	$\mathit{node}_i^{q'_1} = (p, n'_d)$	$\mathit{node}_i^{q'_2} = (p, n_r)$

14.2 When $\mathit{level}_i^{q_1} = (p_l, l) \in Pr$ and $l \in C$

14.2.1 When $\mathit{node}_i^{q_1} = n_d \in C$

Listing 171: Cases in which o results in an empty update set when applied to q_1

```
n_d = n_r or n_r ∉ λ(l)
```

Listing 172: Cases in which o' results in an empty update set when applied to q'_1

```
n_d = n_r or n_r ∉ λ(l)
```

Table 35

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = n_d$	$\mathit{node}_i^{q_2} = n_r$
Trace q'	$\mathit{node}_i^{q'_1} = n_d$	$\mathit{node}_i^{q'_2} = n_r$

14.2.2 When $\mathit{node}_i^{q_1} = (p, n_d) \in Pr$ where $n_d \in C$

Listing 173: Cases in which o results in an empty update set when applied to q_1

```
n_r ∉ λ(l) or n_r ∉ dom_p(p)
```

Listing 174: Cases in which o' results in an empty update set when applied to q'_1

```
n_r ∉ λ(l) or n_r ∉ dom_p(p)
```

Table 36

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p, n_d)$	$\mathit{node}_i^{q_2} = (p, n_r)$
Trace q'	$\mathit{node}_i^{q'_1} = (p, n_d)$	$\mathit{node}_i^{q'_2} = (p, n_r)$

14.2.3 When $\textit{node}_i^{q_1} = (p, ?) \in Pr$

Let p be bound to a value n'_d in q'_1.

Listing 175: Cases in which o results in an empty update set when applied to q_1

```
n_r ∉ λ(l) or n_r ∉ dom_p(p)
```

Listing 176: Cases in which o' results in an empty update set when applied to q'_1

```
n_r ∉ λ(l) or n_r ∉ dom_p(p)
```

Table 37

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = (p, ?)$	$\textit{node}_i^{q_2} = (p, n_r)$
Trace q'	$\textit{node}_i^{q'_1} = (p, n'_d)$	$\textit{node}_i^{q'_2} = (p, n_r)$

14.3 When $\textit{level}_i^{q_1} = (p_l, ?) \in Pr$

14.3.1 When $\textit{node}_i^{q_1} = n \in C$

Not possible to have unbound level and a constant node.

14.3.2 When $\textit{node}_i^{q_1} = (p_n, n) \in Pr$ where $n \in C$

Not possible to have unbound level and an actual constant node.

14.3.3 When $\textit{node}_i^{q_1} = (p_n, ?) \in Pr$

Not possible. Operation will always result in empty update set on schema level since the dice level is unknown.

15 Reset Dice Node (1)

$o = \textsc{ResetDiceNode}_i(p_r, n_r)$ and $o' = \textsc{ResetDiceNode}_i(p_r, n'_r)$.

15.1 When $\textit{level}_i^{q_1} = l \in C$

15.1.1 When $\textit{node}_i^{q_1} = n_d \in C$

When $n_r \in C$ First case.

Listing 177: Cases in which o results in an empty update set when applied to q_1

$n_r \notin \lambda(l)$ **or** $n_r \notin dom_{\mathtt{p}}(p_r)$

Listing 178: Cases in which o' results in an empty update set when applied to q'_1

$n_r \notin \lambda(l)$ **or** $n_r \notin dom_{\mathtt{p}}(p_r)$

Table 38

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = n_d$	$\textit{node}_i^{q_2} = (p_r, n_r)$
Trace q'	$\textit{node}_i^{q'_1} = n_d$	$\textit{node}_i^{q'_2} = (p_r, n_r)$

When $n_r =?$ Second case.

Listing 179: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 180: Cases in which o' results in an empty update set when applied to q'_1

$n'_r \notin \lambda(l)$ **or** $n'_r \notin dom_{\mathtt{p}}(p_r)$

Table 39

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = n_d$	$\textit{node}_i^{q_2} = (p_r, ?)$
Trace q'	$\textit{node}_i^{q'_1} = n_d$	$\textit{node}_i^{q'_2} = (p_r, n'_r)$

Listing 181: Proof binding function

$\pi_{\textit{node}} = \{p_r \mapsto n'_r\}$

15.1.2 When $\textit{node}_i^{q_1} = (p, n_d) \in Pr$ where $n_d \in C$

When $n_r \in C$ First case.

Listing 182: Cases in which o results in an empty update set when applied to q_1

$n_r \notin \lambda(l)$ **or** $n_r \notin dom_{\mathtt{p}}(p_r)$

Listing 183: Cases in which o' results in an empty update set when applied to q_1'

$n_r \notin \lambda(l)$ **or** $n_r \notin dom_{\mathtt{p}}(p_r)$

Table 40

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = (p, n_d)$	$\textit{node}_i^{q_2} = (p_r, n_r)$
Trace q'	$\textit{node}_i^{q_1'} = (p, n_d)$	$\textit{node}_i^{q_2'} = (p_r, n_r)$

When $n_r =?$ Second case.

Listing 184: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 185: Cases in which o' results in an empty update set when applied to q_1'

$n_r' \notin \lambda(l)$ **or** $n_r' \notin dom_{\mathtt{p}}(p_r)$

Table 41

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = (p, n_d)$	$\textit{node}_i^{q_2} = (p_r, ?)$
Trace q'	$\textit{node}_i^{q_1'} = (p, n_d)$	$\textit{node}_i^{q_2'} = (p_r, n_r')$

Listing 186: Proof binding function

$\pi_{\textit{node}} = \{p_r \mapsto n_r'\}$

15.1.3 When $\textit{node}_i^{q_1} = (p, ?) \in Pr$

Let p be bound to a value n_d' in q_1'.

When $n_r \in C$ First case.

Listing 187: Cases in which o results in an empty update set when applied to q_1

$n_r \notin \lambda(l)$ **or** $n_r \notin dom_{\mathtt{p}}(p_r)$

Listing 188: Cases in which o' results in an empty update set when applied to q_1'

```
$n_r \notin \lambda(l)$ or $n_r \notin dom_{\mathfrak{p}}(p_r)$
```

Table 42

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p, ?\,)$	$\mathit{node}_i^{q_2} = (p_r, n_r)$
Trace q'	$\mathit{node}_i^{q_1'} = (p, n_d')$	$\mathit{node}_i^{q_2'} = (p_r, n_r)$

When $n_r =?$ Second case.

Listing 189: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 190: Cases in which o' results in an empty update set when applied to q_1'

```
$n_r' \notin \lambda(l)$ or $n_r' \notin dom_{\mathfrak{p}}(p_r)$
```

Table 43

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p, ?\,)$	$\mathit{node}_i^{q_2} = (p_r, ?\,)$
Trace q'	$\mathit{node}_i^{q_1'} = (p, n_d')$	$\mathit{node}_i^{q_2'} = (p_r, n_r')$

Listing 191: Proof binding function

```
$\pi_{\mathit{node}} = \{p_r \mapsto n_r'\}$
```

15.2 When $\mathit{level}_i^{q_1} = (p_l, l) \in Pr$ and $l \in C$

15.2.1 When $\mathit{node}_i^{q_1} = n_d \in C$

When $n_r \in C$ First case.

Listing 192: Cases in which o results in an empty update set when applied to q_1

```
$n_r \notin \lambda(l)$ or $n_r \notin dom_{\mathfrak{p}}(p_r)$
```

Listing 193: Cases in which o' results in an empty update set when applied to q_1'

```
$n_r \notin \lambda(l)$ or $n_r \notin dom_{\mathfrak{p}}(p_r)$
```

Table 44

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = n_d$	$\mathit{node}_i^{q_2} = (p_r, n_r)$
Trace q'	$\mathit{node}_i^{q'_1} = n_d$	$\mathit{node}_i^{q'_2} = (p_r, n_r)$

When $n_r = ?$ Second case.

Listing 194: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 195: Cases in which o' results in an empty update set when applied to q'_1

$n'_r \notin \lambda(l)$ **or** $n'_r \notin dom_{\mathtt{p}}(p_r)$

Table 45

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = n_d$	$\mathit{node}_i^{q_2} = (p_r, ?)$
Trace q'	$\mathit{node}_i^{q'_1} = n_d$	$\mathit{node}_i^{q'_2} = (p_r, n'_r)$

Listing 196: Proof binding function

$\pi_{\mathit{node}} = \{p_r \mapsto n'_r\}$

15.2.2 When $\mathit{node}_i^{q_1} = (p, n_d) \in Pr$ where $n \in C$

When $n_r \in C$ First case.

Listing 197: Cases in which o results in an empty update set when applied to q_1

$n_r \notin \lambda(l)$ **or** $n_r \notin dom_{\mathtt{p}}(p_r)$

Listing 198: Cases in which o' results in an empty update set when applied to q'_1

$n_r \notin \lambda(l)$ **or** $n_r \notin dom_{\mathtt{p}}(p_r)$

Table 46

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p, n_d)$	$\mathit{node}_i^{q_2} = (p_r, n_r)$
Trace q'	$\mathit{node}_i^{q'_1} = (p, n_d)$	$\mathit{node}_i^{q'_2} = (p_r, n_r)$

When $n_r = ?$ Second case.

Listing 199: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 200: Cases in which o' results in an empty update set when applied to q_1'

$n_r' \notin \lambda(l)$ **or** $n_r' \notin dom_{\mathtt{p}}(p_r)$

Table 47

	Source Situation 1	Target Situation 2
Schema q	$node_i^{q_1} = (p, n_d)$	$node_i^{q_2} = (p_r, ?)$
Trace q'	$node_i^{q_1'} = (p, n_d)$	$node_i^{q_2'} = (p_r, n_r')$

Listing 201: Proof binding function

$\pi_{node} = \{p_r \mapsto n_r'\}$

15.2.3 When $node_i^{q_1} = (p, ?) \in Pr$

Let p be bound to a value n_d' in q_1'.

When $n_r \in C$ First case.

Listing 202: Cases in which o results in an empty update set when applied to q_1

$n_r \notin \lambda(l)$ **or** $n_r \notin dom_{\mathtt{p}}(p_r)$

Listing 203: Cases in which o' results in an empty update set when applied to q_1'

$n_r \notin \lambda(l)$ **or** $n_r \notin dom_{\mathtt{p}}(p_r)$

Table 48

	Source Situation 1	Target Situation 2
Schema q	$node_i^{q_1} = (p, ?)$	$node_i^{q_2} = (p_r, n_r)$
Trace q'	$node_i^{q_1'} = (p, n_d')$	$node_i^{q_2'} = (p_r, n_r)$

When $n_r = ?$ Second case.

Listing 204: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 205: Cases in which o' results in an empty update set when applied to q'_1

```
n'_r ∉ λ(l) or n'_r ∉ dom_p(p_r)
```

Table 49

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = (p, ?)$	$\textit{node}_i^{q_2} = (p_r, ?)$
Trace q'	$\textit{node}_i^{q'_1} = (p, n'_d)$	$\textit{node}_i^{q'_2} = (p_r, n'_r)$

Listing 206: Proof binding function

```
π_node = {p_r ↦ n'_r}
```

15.3 When $\textit{level}_i^{q_1} = (p_l, ?) \in Pr$

15.3.1 When $\textit{node}_i^{q_1} = n \in C$

Not possible to have unbound level and a constant node.

15.3.2 When $\textit{node}_i^{q_1} = (p_n, n) \in Pr$ where $n \in C$

Not possible to have unbound level and a constant actual node.

15.3.3 When $\textit{node}_i^{q_1} = (p_n, ?) \in Pr$

When $n_r \in C$ Not possible to have unbound level and assign a constant node as this scenario is prevented per the definition of the operation.

When $n_r = ?$ Second case.

Listing 207: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 208: Cases in which o' results in an empty update set when applied to q'_1

```
n'_r ∉ λ(l') or n'_r ∉ dom_p(p_r)
```

Table 50

	Source Situation 1	Target Situation 2
Schema q	$\textit{level}_i^{q_1} = (p_l, ?)$, $\textit{node}_i^{q_1} = (p, ?)$	$\textit{level}_i^{q_1} = (p_l, ?)$ $\textit{node}_i^{q_2} = (p_r, ?)$
Trace q'	$\textit{level}_i^{q_1} = (p_l, l')$, $\textit{node}_i^{q'_1} = (p, n'_d)$	$\textit{level}_i^{q_1} = (p_l, l')$ $\textit{node}_i^{q'_2} = (p_r, n'_r)$

Listing 209: Proof binding function

```
π_node = {p_l ↦ l', p_r ↦ n'_r}
```

$$\pi_{\mathsf{node}} = \{p_l \mapsto l', p_r \mapsto n_r'\}$$

16 Reset Dice Node (2)

$o = \textsc{ResetDiceNode}_i(n_r)$ and $o' = \textsc{ResetDiceNode}_i(n_r)$.

16.1 When $\textit{level}_i^{q_1} = l \in C$

16.1.1 When $\textit{node}_i^{q_1} = n_d \in C$

Listing 210: Cases in which o results in an empty update set when applied to q_1

$n_d = n_r$ **or** $n_r \notin \lambda(l)$

Listing 211: Cases in which o' results in an empty update set when applied to q'_1

$n_d = n_r$ **or** $n_r \notin \lambda(l)$

Table 51

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = n_d$	$\textit{node}_i^{q_2} = n_r$
Trace q'	$\textit{node}_i^{q'_1} = n_d$	$\textit{node}_i^{q'_2} = n_r$

16.1.2 When $\textit{node}_i^{q_1} = (p, n_d) \in Pr$ where $n \in C$

Listing 212: Cases in which o results in an empty update set when applied to q_1

$n_r \notin \lambda(l)$

Listing 213: Cases in which o' results in an empty update set when applied to q'_1

$n_r \notin \lambda(l)$

Table 52

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = (p, n_d)$	$\textit{node}_i^{q_2} = n_r$
Trace q'	$\textit{node}_i^{q'_1} = (p, n_d)$	$\textit{node}_i^{q'_2} = n_r$

16.1.3 When $\textit{node}_i^{q_1} = (p, ?) \in Pr$

Let p be bound to a value n'_d in q'_1.

Listing 214: Cases in which o results in an empty update set when applied to q_1

$n_r \notin \lambda(l)$

Listing 215: Cases in which o' results in an empty update set when applied to q'_1

```
n_r ∉ λ(l)
```

Table 53

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p, ?\,)$	$\mathit{node}_i^{q_2} = n_r$
Trace q'	$\mathit{node}_i^{q'_1} = (p, n'_d)$	$\mathit{node}_i^{q'_2} = n_r$

16.2 When $\mathit{level}_i^{q_1} = (p_l, l) \in Pr$ and $l \in C$

16.2.1 When $\mathit{node}_i^{q_1} = n_d \in C$

Listing 216: Cases in which o results in an empty update set when applied to q_1

```
n_d = n_r or n_r ∉ λ(l)
```

Listing 217: Cases in which o' results in an empty update set when applied to q'_1

```
n_d = n_r or n_r ∉ λ(l)
```

Table 54

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = n_d$	$\mathit{node}_i^{q_2} = n_r$
Trace q'	$\mathit{node}_i^{q'_1} = n_d$	$\mathit{node}_i^{q'_2} = n_r$

16.2.2 When $\mathit{node}_i^{q_1} = (p, n_d) \in Pr$ where $n \in C$

Listing 218: Cases in which o results in an empty update set when applied to q_1

```
n_r ∉ λ(l)
```

Listing 219: Cases in which o' results in an empty update set when applied to q'_1

```
n_r ∉ λ(l)
```

Table 55

	Source Situation 1	Target Situation 2
Schema q	$\mathit{node}_i^{q_1} = (p, n_d)$	$\mathit{node}_i^{q_2} = n_r$
Trace q'	$\mathit{node}_i^{q'_1} = (p, n_d)$	$\mathit{node}_i^{q'_2} = n_r$

16.2.3 When $\textit{node}_i^{q_1} = (p, ?\,) \in Pr$

Let p be bound to a value n'_d in q'_1.

Listing 220: Cases in which o results in an empty update set when applied to q_1

$n_r \notin \lambda(l)$

Listing 221: Cases in which o' results in an empty update set when applied to q'_1

$n_r \notin \lambda(l)$

Table 56

	Source Situation 1	Target Situation 2
Schema q	$\textit{node}_i^{q_1} = (p, ?\,)$	$\textit{node}_i^{q_2} = n_r$
Trace q'	$\textit{node}_i^{q'_1} = (p, n'_d)$	$\textit{node}_i^{q'_2} = n_r$

16.3 When $\textit{level}_i^{q_1} = (p_l, ?\,) \in Pr$

16.3.1 When $\textit{node}_i^{q_1} = n \in C$

Not possible to have unbound level and a constant node.

16.3.2 When $\textit{node}_i^{q_1} = (p_n, n) \in Pr$ where $n \in C$

Not possible to have unbound level and a constant node.

16.3.3 When $\textit{node}_i^{q_1} = (p_n, ?\,) \in Pr$

Not possible to have unbound level and assign a constant node as this scenario is prevented per the definition of the operation.

17 Change Level To

r and d in level names denote new and old, respectively (not up and down as in previous section).

$o = \textsc{ChangeLevelTo}_i(l_r)$ and $o' = \textsc{ChangeLevelTo}_i(l_r)$.

17.1 When $\mathit{level}_i^{q_1} = l_d \in C$

Listing 222: Cases in which o results in an empty update set when applied to q_1

```
l_d = l_r
```

Listing 223: Cases in which o' results in an empty update set when applied to q'_1

```
l_d = l_r
```

Table 57

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = l_d$	$\mathit{level}_i^{q_2} = l_r$
Trace q'	$\mathit{level}_i^{q'_1} = l_d$	$\mathit{level}_i^{q'_2} = l_r$

17.2 When $\mathit{level}_i^{q_1} = (p, l_d) \in Pr$ where $l_d \in C$

Listing 224: Cases in which o results in an empty update set when applied to q_1

```
l_r ∉ dom_p(p)
```

Listing 225: Cases in which o' results in an empty update set when applied to q'_1

```
l_r ∉ dom_p(p)
```

Table 58

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = (p, l_d)$	$\mathit{level}_i^{q_2} = (p, l_r)$
Trace q'	$\mathit{level}_i^{q'_1} = (p, l_d)$	$\mathit{level}_i^{q'_2} = (p, l_r)$

17.3 When $\mathit{level}_i^{q_1} = (p, ?) \in Pr$

Let q'_1 bind p to l'_d.

Listing 226: Cases in which o results in an empty update set when applied to q_1

```
l_r ∉ dom_p(p)
```

Listing 227: Cases in which o' results in an empty update set when applied to q'_1

$l_r \notin dom_{\mathtt{p}}(p)$

Table 59

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = (p, ?\,)$	$\mathit{level}_i^{q_2} = (p, l_r)$
Trace q'	$\mathit{level}_i^{q'_1} = (p, l'_d)$	$\mathit{level}_i^{q'_2} = (p, l_r)$

18 Reset Dice Level (1)

r and d in level names denote new and old, respectively (not up and down as in previous section).

$o = \textsc{ResetDiceLevel}_i(p_r, l_r)$ and $o' = \textsc{ResetDiceLevel}_i(p_r, l'_r)$.

18.1 When $l_r \in C$

18.1.1 When $\mathit{level}_i^{q_1} = l_d \in C$

Listing 228: Cases in which o results in an empty update set when applied to q_1

$l_r \notin dom_{\mathfrak{p}}(p_r)$

Listing 229: Cases in which o' results in an empty update set when applied to q'_1

$l_r \notin dom_{\mathfrak{p}}(p_r)$

Table 60

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = l_d$	$\mathit{level}_i^{q_2} = (p_r, l_r)$
Trace q'	$\mathit{level}_i^{q'_1} = l_d$	$\mathit{level}_i^{q'_2} = (p_r, l_r)$

18.1.2 When $\mathit{level}_i^{q_1} = (p, l_d) \in Pr$ where $l_d \in C$

Listing 230: Cases in which o results in an empty update set when applied to q_1

$l_r \notin dom_{\mathfrak{p}}(p_r)$

Listing 231: Cases in which o' results in an empty update set when applied to q'_1

$l_r \notin dom_{\mathfrak{p}}(p_r)$

Table 61

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = (p, l_d)$	$\mathit{level}_i^{q_2} = (p_r, l_r)$
Trace q'	$\mathit{level}_i^{q'_1} = (p, l_d)$	$\mathit{level}_i^{q'_2} = (p_r, l_r)$

18.1.3 When $\mathit{level}_i^{q_1} = (p, ?) \in Pr$

Let q'_1 bind p to l'_d.

Listing 232: Cases in which o results in an empty update set when applied to q_1

$l_r \notin dom_{\mathfrak{p}}(p_r)$

Listing 233: Cases in which o' results in an empty update set when applied to q'_1

$l_r \notin dom_{\mathfrak{p}}(p_r)$

Table 62

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = (p, ?)$	$\mathit{level}_i^{q_2} = (p_r, l_r)$
Trace q'	$\mathit{level}_i^{q'_1} = (p, l'_d)$	$\mathit{level}_i^{q'_2} = (p_r, l_r)$

18.2 When $l_r = ?$

18.2.1 When $\mathit{level}_i^{q_1} = l_d \in C$

Let π_o bind p_r to l'_r.

Listing 234: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 235: Cases in which o' results in an empty update set when applied to q'_1

$l'_r \notin dom_{\mathfrak{p}}(p_r)$

Table 63

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = l_d$	$\mathit{level}_i^{q_2} = (p_r, ?)$
Trace q'	$\mathit{level}_i^{q'_1} = l_d$	$\mathit{level}_i^{q'_2} = (p_r, l'_r)$

Listing 236: Proof binding function

$\pi_{\mathit{level}} = \{p_r \mapsto l'_r\}$

18.2.2 When $\mathit{level}_i^{q_1} = (p, l_d) \in Pr$ where $l_d \in C$

Listing 237: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 238: Cases in which o results in an empty update set when applied to q'_1

$l'_r \notin dom_{\mathfrak{p}}(p_r)$

Listing 239: Proof binding function

$\pi_{\mathit{level}} = \{p_r \mapsto l'_r\}$

Table 64

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = (p, l_d)$	$\mathit{level}_i^{q_2} = (p_r, ?)$
Trace q'	$\mathit{level}_i^{q_1'} = (p, l_d)$	$\mathit{level}_i^{q_2'} = (p_r, l_r')$

18.2.3 When $\mathit{level}_i^{q_1} = (p, ?) \in Pr$

Let q_1' bind p to l_d'.

Listing 240: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 241: Cases in which o results in an empty update set when applied to q_1'

$l_r' \notin dom_{\mathfrak{p}}(p_r)$

Table 65

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = (p, ?)$	$\mathit{level}_i^{q_2} = (p_r, ?)$
Trace q'	$\mathit{level}_i^{q_1'} = (p, l_d')$	$\mathit{level}_i^{q_2'} = (p_r, l_r')$

Listing 242: Proof binding function

$\pi_{\mathit{level}} = \{p_r \mapsto l_r'\}$

19 Reset Dice Level (2)

r and d in level names denote new and old, respectively.

$o = \textsc{ResetDiceLevel}_i(l_r)$ and $o' = \textsc{ResetDiceLevel}_i(l_r)$.

19.1 When $\mathit{level}_i^{q_1} = l_d \in C$

Listing 243: Cases in which o results in an empty update set when applied to q_1

$l_d = l_r$

Listing 244: Cases in which o' results in an empty update set when applied to q_1'

$l_d = l_r$

Table 66

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = l_d$	$\mathit{level}_i^{q_2} = l_r$
Trace q'	$\mathit{level}_i^{q_1'} = l_d$	$\mathit{level}_i^{q_2'} = l_r$

19.2 When $\mathit{level}_i^{q_1} = (p, l_d) \in Pr$ where $l_d \in C$

Listing 245: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 246: Cases in which o' results in an empty update set when applied to q_1'

```
None
```

Table 67

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = (p, l_d)$	$\mathit{level}_i^{q_2} = l_r$
Trace q'	$\mathit{level}_i^{q_1'} = (p, l_d)$	$\mathit{level}_i^{q_2'} = l_r$

19.3 When $\mathit{level}_i^{q_1} = (p, ?\,) \in Pr$

Let q_1' bind p to l_d'.

Listing 247: Cases in which o results in an empty update set when applied to q_1

```
None
```

Listing 248: Cases in which o' results in an empty update set when applied to q'_1

```
None
```

Table 68

	Source Situation 1	Target Situation 2
Schema q	$\mathit{level}_i^{q_1} = (p, ?\,)$	$\mathit{level}_i^{q_2} = l_r$
Trace q'	$\mathit{level}_i^{q'_1} = (p, l'_d)$	$\mathit{level}_i^{q'_2} = l_r$

20 Move to Level and Node (1+2+3)

Since the two operation types – involved in each one of these three MoveToLevelAndNode$_i$ operation types – are semantics-preserving, the semantics preservation for each of the three MoveToLevelAndNode$_i$ operations types follows directly from the definition of the operation type.