

# fn make\_is\_equal

Sílvia Casacuberta

This proof resides in “**contrib**” because it has not completed the vetting process.

Proves soundness of `fn make_is_equal` in `mod.rs` at [commit 0db9c6036](#) (outdated<sup>1</sup>). The transformation checks if each element in a vector dataset is equivalent to a given value, returning a vector of booleans.

## Vetting History

- [Pull Request #512](#)

## 1 Hoare Triple

### Precondition

To ensure the correctness of the output, we require the following preconditions:

- Type `TIA` must have trait `PartialEq`.
- Type `M` must have trait `DatasetMetric`.
- `MetricSpace` is implemented for `(VectorDomain<AtomDomain<TIA>>, M)`. Therefore `M` is a valid metric on `VectorDomain<AtomDomain<TIA>>`.
- `MetricSpace` is implemented for `(VectorDomain<AtomDomain<bool>>, M)`.

### Pseudocode

```
1 def make_is_equal(  
2     input_domain: VectorDomain[AtomDomain[TIA]],  
3     input_metric: M,  
4     value: TIA  
5 ): #  
6     output_row_domain = atom_domain(T=bool)  
7  
8     def is_equal(arg: TA) -> TA: #  
9         return value == arg  
10  
11     return make_row_by_row( #  
12         input_domain,  
13         input_metric,  
14         output_row_domain,  
15         is_equal  
16 )
```

<sup>1</sup>See new changes with `git diff 0db9c6036..8753487 rust/src/transformations/manipulation/mod.rs`

## Postconditions

**Theorem 1.1.** For every setting of the input parameters (`input_domain`, `input_metric`, `value`) to `make_is_equal` such that the given preconditions hold, `make_is_equal` raises an error (at compile time or run time) or returns a valid transformation. A valid transformation has the following properties:

1. (Data-independent runtime errors). For every pair of members  $x$  and  $x'$  in `input_domain`, `invoke(x)` and `invoke(x')` either both return the same error or neither return an error.
2. (Appropriate output domain). For every member  $x$  in `input_domain`, `function(x)` is in `output_domain` or raises a data-independent runtime error.
3. (Stability guarantee). For every pair of members  $x$  and  $x'$  in `input_domain` and for every pair  $(d\_in, d\_out)$ , where `d_in` has the associated type for `input_metric` and `d_out` has the associated type for `output_metric`, if  $x, x'$  are `d_in`-close under `input_metric`, `stability_map(d_in)` does not raise an error, and `stability_map(d_in) = d_out`, then `function(x), function(x')` are `d_out`-close under `output_metric`.

## 2 Proof

**Lemma 2.1.** The invocation of `make_row_by_row` (line 11) satisfies its preconditions.

*Proof.* The preconditions of `make_is_equal` and pseudocode definition (line 5) ensure that the type preconditions of `make_row_by_row` are satisfied. The remaining preconditions of `make_row_by_row` are:

- `row_function` has no side-effects.
- If the input to `row_function` is a member of `input_domain`'s row domain, then the output is a member of `output_row_domain`.

The first precondition is satisfied by the definition of `is_equal` (line 8) in the pseudocode.

For the second precondition, assume the input is a member of `input_domain`'s row domain. By the definition of `PartialEq`, the output of `is_equal` is boolean, and `AtomDomain<bool>` includes all booleans. Therefore, the output is a member of `output_row_domain`.  $\square$

We now prove the postcondition of `make_is_equal`.

*Proof.* By 2, the preconditions of `make_row_by_row` are satisfied. Thus, by the definition of `make_row_by_row`, the output is a valid transformation.  $\square$