# fn make_privacy_filter

### Michael Shoemate

> This proof resides in **"contrib"** because it has not completed the vetting process.

Proves soundness of fn make_privacy_filter.

## 1   Hoare Triple

### Precondition

**Compiler-verified**

- Argument odometer of type Odometer<DI, MI, MO, Q, A>.

- Argument d_in of type MI_Distance, the associated distance type of MI.

- Argument d_out of type MO_Distance, the associated distance type of MO.

- Generic DI implements Domain.

- Generic MI implements Metric.

- Generic MO implements Measure.

- MI_Distance implements ProductOrd.

- MO_Distance implements ProductOrd.

- (DI, MI) implements MetricSpace.

**User-verified**

None

### Pseudocode

```
1  def make_privacy_filter(
2      odometer: Odometer[DI, MI, MO, Q, A],
3      d_in: MI_Distance,
4      d_out: MO_Distance,
5  ) -> Measurement[DI, OdometerQueryable[MI, MO, Q, A], MI, MO]:
6      odo_function = odometer.function
7
8      #
9      continuation_rule = new_continuation_rule(d_in, d_out, Q, A, MI_Distance, MO_Distance)
10     def function(arg: DI_Carrier) -> OdometerQueryable[MI, MO, Q, A]:
11         return wrap(continuation_rule, lambda: odo_function.eval(arg))  #
12
13     def privacy_map(d_in_p: MI_Distance) -> MO_Distance:
```

```
14          if d_in_p.total_gt(d_in):
15              raise "input distance must not be greater than d_in"
16
17          return d_out
18
19      return Measurement.new(
20          odometer.input_domain,
21          Function.new_interactive(function),
22          odometer.input_metric,
23          odometer.output_measure,
24          PrivacyMap.new_fallible(privacy_map),
25      )
```

**Postcondition**

For every setting of the input parameters (`odometer, d_out, DI, MI, MO, Q, A`) to `make_privacy_filter` such that the given preconditions hold, `make_privacy_filter` raises an error (at compile time or run time) or returns a valid odometer. A valid odometer 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. (Valid odometer queryable). For every member $x$ in `input_domain`, where `function`$(x)$ does not raise an error, `function`$(x)$ returns a valid odometer queryable.

*Proof of data-independent errors.* `Function`.`eval` on line 11 has data-independent exceptions, because the function is from `odometer`, which is a valid odometer. Since this is the only location where an exception can be raised, the data-independent errors property holds. □

*Proof of privacy guarantee.* By the definition of a valid odometer queryable, the output of `make_privacy_filter` upholds the privacy guarantee. □