fn make_geometric

Michael Shoemate

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

Proves soundness of the implementation of `make_geometric` in `mod.rs` at commit f5bb719 (outdated[1]).

The implementation of this function constructs a random variable denoting the noise distribution to add, and then dispatches to the `MakeNoise<DI, MI, MO>` trait which constructs the core mechanism and wraps it in pre-processing transformations and post-processors to match the desired parameterization.

1 Hoare Triple

Precondition

Compiler-Verified

- generic `DI` implements trait `Domain`
- generic `MI` implements trait `Metric`
- generic `MO` implements trait `Measure`
- type `DiscreteLaplace` implements trait `MakeNoise<DI, MI, MO>`
- type `ConstantTimeGeometric` implements trait `MakeNoise<DI, MI, MO>` These traits constrain the choice of input domain, input metric and output measure to those that can form valid measurements when adding noise from these distributions.
- type `(DI, MI)` implements trait `MetricSpace`

User-Verified

None

Pseudocode

```
def make_geometric(
    input_domain: DI,
    input_metric: MI,
    scale: f64,
    bounds: Option[tuple[DI_Atom, DI_Atom]],
) -> Measurement[DI, DI_Carrier, MI, MO]:
    input_space = input_domain, input_metric
    if bounds is None:
        return DiscreteLaplace(scale, k=None).make_noise(input_space)
    else:
        return ConstantTimeGeometric(scale, bounds).make_noise(input_space)
```

[1] See new changes with `git diff f5bb719..e1ce697b rust/src/measurements/noise/distribution/geometric/mod.rs`

Postcondition

Theorem 1.1.

Theorem 1.2. For every setting of the input parameters (`input_domain, input_metric, scale, bounds, DI, MI, MO`) to `make_geometric` such that the given preconditions hold, `make_geometric` raises an exception (at compile time or run time) or returns a valid measurement. A valid measurement has the following property:

1. (Privacy guarantee). For every pair of elements x, 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_measure`, if x, x' are `d_in`-close under `input_metric`, `privacy_map(d_in)` does not raise an exception, and `privacy_map(d_in)` $\leq$ `d_out`, then `function`(x), `function`(x') are `d_out`-close under `output_measure`.

Proof. If bounds are supplied, this constructor builds a specialized mechanism that adds noise to the input data from the `ConstantTimeGeometric` random variable. Otherwise noise is added from the `DiscreteLaplace` random variable, which uses a logarithmic-time discrete laplace sampling algorithm.

Since `MakeNoise.make_noise` has no preconditions, the postcondition follows, which matches the postcondition for this function. □