MakeNoise<VectorDomain<AtomDomain<IBig», MI, MO> for RV

Michael Shoemate

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

Proves soundness of the implementation of `MakeNoise` for `RV` over vectors of big integers in `mod.rs` at commit f5bb719 (outdated[1]).

This is the core implementation of all variations of the gaussian or laplace mechanism.

1 Hoare Triple

Precondition

Compiler-Verified

`MakeNoise` is parameterized as follows:

- `MI` implements trait `Metric`
- `MO` implements trait `Measure`
- `RV` implements trait `Sample`

The following trait bounds are also required:

- `(VectorDomain<AtomDomain<IBig>>, MI)` implements trait `MetricSpace`
- `RV` implements `NoisePrivacyMap<MI, MO>`

User-Verified

None

Pseudocode

```
# analogous to impl MakeNoise<VectorDomain<AtomDomain<IBig>>, MI, MO> for RV in Rust
class RV:
    def make_noise(self, input_space) -> Measurement[VectorDomain[AtomDomain[IBig]], Vec[IBig], MI, MO]:
        input_domain, input_metric = input_space
        return Measurement.new(
            input_domain,
            Function.new_fallible(
                lambda x: [self.sample(x_i) for x_i in x]), #
            input_metric,
```

[1] See new changes with `git diff f5bb719..7c365e46 rust/src/measurements/noise/mod.rs`

```
10              MO.default(),
11              self.noise_privacy_map(),  #
12          )
```

Postcondition

Theorem 1.1.

Theorem 1.2. For every setting of the input parameters (`self, input_space, MI, MO, RV`) to `make_noise` such that the given preconditions hold, `make_noise` 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 of data-independent errors. The precondition of `Sample.sample` requires that `self` is a valid distribution. This is satisfied by the postcondition of `NoisePrivacyMap<MI, MO>` on line 11. The postcondition of `Sample.sample` guarantees that the function only ever returns an error independently of the data. □

For the proof of the privacy guarantee, start by reviewing the postcondition of `NoisePrivacyMap<MI, MO>`, which has an associated function `noise_privacy_map` to be called on line 11.

Lemma 1.3 (Postcondition of `NoisePrivacyMap`)**.** Given a distribution `self`, returns `Err(e)` if `self` is not a valid distribution. Otherwise the output is `Ok(privacy_map)` where `privacy_map` observes the following:

Define `function(x) = x + Z` where `Z` is a vector of iid samples from `self`.

For every pair of elements x, x' in `VectorDomain<AtomDomain<IBig>>`, 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 of privacy guarantee. Assuming line 11 does not fail, then the returned privacy map is subject to Theorem 1.3. The privacy guarantee applies when `function(x) = x + Z`, where `Z` is a vector of iid samples from `self`. In this case `self` describes the noise distribution.

We argue that `function` is consistent with the function described in Lemma 1.3. Line 8 calls `self.sample(x_i)` on each element in the input vector. The precondition that `self` represents a valid distribution is satisfied by the postcondition of Lemma 1.3; the distribution is valid when the construction of the privacy map does not raise an exception. Since the preconditions for `Sample.sample` are satisfied, the postcondition claims that either an error is returned independently of the input `shift`, or `shift + Z` where `Z` is a sample from the distribution defined by `self`. This is consistent with the definition of `function(x)` in the privacy map.

Therefore, the privacy guarantee from Lemma 1.3 applies to the returned measurement. □