fn make_float_to_bigint_threshold

Michael Shoemate

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

Proves soundness of the implementation of `make_float_to_bigint_threshold` in `mod.rs` at commit f5bb719 (outdated[1]).

1 Hoare Triple

Precondition

Compiler-Verified

- Generic `TK` implements trait `Hashable`
- Generic `TV` implements trait `Float`
- Const-generic `P` is of type `usize`
- Generic `QI` implements trait `Number`
- Type `RBig` implements traits `TryFrom<TV>` and `TryFrom<QI>`. This is for fallible exact casting to rationals from floats in the function and input sensitivity in the privacy map.
- Type `i32` implements trait `ExactIntCast<T as FloatBits>::Bits>`, This requirement means that the raw bits of `T` can be exactly cast to an `i32`.

User-Verified

None

Pseudocode

```python
def make_float_to_bigint_threshold(
    input_space: tuple[
        MapDomain[AtomDomain[TK], MapDomain[TV]], L0PInfDistance[P, AbsoluteDistance[QI]]
    ],
    threshold: TV,
    k: i32,
) -> Transformation[
    MapDomain[AtomDomain[TK], AtomDomain[TV]],
    MapDomain[AtomDomain[TK], AtomDomain[IBig]],
    L0PInfDistance[P, AbsoluteDistance[QI]],
    L0PInfDistance[P, AbsoluteDistance[RBig]],
]:
    input_domain, input_metric = input_space
```

[1] See new changes with `git diff f5bb719..e78d6c1 rust/src/measurements/noise_threshold/nature/float/mod.rs`

```
14     if input_domain.value_domain.nan():
15         raise "input_domain hashmap values may not contain NaN elements"
16 
17     r_threshold = RBig.try_from(threshold)
18 
19     min_k = get_min_k(TV)
20     if k < min_k:  #
21         raise f"k ({k}) must not be smaller than {min_k}"
22 
23     def value_function(val):  #
24         try:  #
25             val = RBig.try_from(val)
26         except Exception:
27             val = RBig.ZERO
28 
29         return find_nearest_multiple_of_2k(val, k)  #
30 
31     def stability_map(d_in):
32         l0, lp, li = d_in
33 
34         r_lp = RBig.try_from(lp)
35         r_lp_round = get_rounding_distance(k, usize.from_(l0), P)
36         r_lp = x_mul_2k(r_lp + r_lp_round, -k)  #
37 
38         r_li = RBig.try_from(li)
39         if r_li > x_mul_2k(r_threshold, -k): #
40             raise f"li ({li}) must not be larger than threshold ({threshold})"
41         r_li_round = get_rounding_distance(k, 1, P)
42         r_li = x_mul_2k(r_li + r_li_round, -k)  #
43 
44         return l0, r_lp, r_li
45 
46     return Transformation.new(
47         input_domain,
48         MapDomain(  #
49             key_domain=input_domain.key_domain,
50             value_domain=AtomDomain.default(IBig),
51         ),
52         Function.new(lambda x: {k: value_function(v) for k, v in x.items()}),
53         input_metric,
54         L0PI.default(),
55         StabilityMap.new_fallible(stability_map),
56     )
```

Postcondition

Theorem 1.1.

Theorem 1.2. For every setting of the input parameters (`input_space, k, TK, TV, P, QI`) to `make_float_to_bigint_thr` such that the given preconditions hold, `make_float_to_bigint_threshold` raises an exception (at compile time or run time) or returns a valid transformation. A valid transformation has the following properties:

1. (Appropriate output domain). For every element x in `input_domain`, `function`(x) is in `output_domain` or raises a data-independent runtime exception.

2. (Stability 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_metric`, if x, x' are `d_in`-close under `input_metric`, `stability_map(d_in)` does not raise an exception, and `stability_map(d_in)` $\leq$ `d_out`, then `function`(x), `function`(x') are `d_out`-close under `output_metric`.

Proof. In the definition of the function on line 23, `RBig.try_from` is infallible when the input is non-nan. The precondition for `find_nearest_multiple_of_2k` is satisfied by line 20, so `find_nearest_multiple_of_2k` is infallible. There are no other sources of error in the function, so the function cannot raise data-dependent errors.

The function also always returns a hashmap with the same keys, and IBig values, meaning the output of the function is always a member of the output domain, as defined on line 48.

The stability argument breaks down into three parts:

- The casting from float to rational on line 24 is 1-stable, because the real values of the numbers remain un-changed, meaning the distance between adjacent inputs always remains the same.

- The rounding on line 29 can cause an increase in the sensitivity equal to $\Delta_0^{1/p}(2^k - 2^{k_{min}})$.

$$\max_{x \sim x'} d_{Lp}(f(x), f(x')) \tag{1}$$
$$= \max_{x \sim x'} |\text{round}_k(x) - \text{round}_k(x')|_p \tag{2}$$
$$\leq \max_{x \sim x'} |(x + 2^{k-1}) - (x' - 2^{k-1} + 2^{k_{min}})|_p \tag{3}$$
$$\leq \max_{x \sim x'} |x - x'|_p + |R \cdot (2^k - 2^{k_{min}})|_p \qquad \text{where } R \in \{0,1\}^n \text{ with weight } \Delta_0 \tag{4}$$
$$= \max_{x \sim x'} d_{Lp}(x, x') + \Delta_0^{1/p}(2^k - 2^{k_{min}}) \tag{5}$$
$$= \texttt{d\_in} + \Delta_0^{1/p}(2^k - 2^{k_{min}}) \tag{6}$$

  This increase in the sensitivity is reflected on line 36, which, by the postcondition of `get_rounding_distance`, returns the maximum increase in sensitivity due to rounding. The rounding distance is added to the L_p sensitivity.

  A similar analysis follows for L_∞ sensitivity on line 42, where only one rounding occurs instead of Δ_0. Notice that the check on line 39 is not necessary for the privacy guarantee, it improves the quality of the error message. The equivalent error raised from the core mechanism privacy map is not as user-friendly, because the constants are scaled by 2^k.

- The discarding of the denominator on line 29 is 2^k-stable, as the denominator is 2^k. This increase in sensitivity is also reflected on lines 36 and 42, where the sensitivity is multiplied by 2^{-k}.

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_metric`, if x, x' are `d_in`-close under `input_metric`, `stability_map`(`d_in`) does not raise an exception, and `stability_map`(`d_in`) $\leq$ `d_out`, then `function`(x), `function`(x') are `d_out`-close under `output_metric`.

□