fn make_float_to_bigint

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Proves soundness of the implementation of `make_float_to_bigint` in `mod.rs` at commit f5bb719 (outdated[1]).

1 Hoare Triple

Precondition

Compiler-Verified

- Generic `T` implements trait `Float`
- Const-generic `P` is of type `usize`
- Generic `QI` implements trait `Number`
- Type `RBig` implements traits `TryFrom<T>` 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

```
def make_float_to_bigint(
    input_space: tuple[VectorDomain[AtomDomain[T]], LpDistance[P, QI]], k: i32
) -> Transformation[
    VectorDomain[AtomDomain[T]],
    VectorDomain[AtomDomain[IBig]],
    LpDistance[P, QI],
    LpDistance[P, RBig],
]:
    input_domain, input_metric = input_space
    if input_domain.element_domain.nullable():
        raise "input_domain may not contain NaN elements"

    size = input_domain.size
    rounding_distance = get_rounding_distance(k, size, T) #

```

[1] See new changes with `git diff f5bb719..f5c7b95 rust/src/measurements/noise/nature/float/mod.rs`

```
16      def elementwise_function(x_i):  #
17          x_i = RBig.try_from(x_i).unwrap_or(RBig.ZERO)  #
18          return find_nearest_multiple_of_2k(x_i, k)  #
19
20      def stability_map(d_in):
21          try:
22              d_in = RBig.try_from(d_in)
23          except Exception:
24              raise f"d_in ({d_in}) must be finite"
25          return x_mul_2k(d_in + rounding_distance, -k)  #
26
27      return Transformation.new(
28          input_domain,
29          VectorDomain(  #
30              element_domain=AtomDomain.default(IBig),
31              size=size,
32          ),
33          Function.new(lambda x: [elementwise_function(x_i) for x_i in x]),
34          input_metric,
35          LpDistance.default(),
36          StabilityMap.new_fallible(stability_map),
37      )
```

Postcondition

Theorem 1.1.

Theorem 1.2. For every setting of the input parameters (`input_space, k, T, P, QI`) to `make_float_to_bigint` such that the given preconditions hold, `make_float_to_bigint` 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`.

Proof. In the definition of the function on line 16, `RBig.try_from` is infallible when the input is non-nan making the function infallible. Line 14 checks that k is not `i32.MIN`, which satisfies the precondition for `find_nearest_multiple_of_2k` on line 18, and ensures that negation is well-defined on line 25. There are no other sources of error in the function, so the function cannot raise data-dependent errors.

The function also always returns a vector of IBigs, of the same length as the input, meaning the output of the function is always a member of the output domain, as defined on line 29.

The stability argument breaks down into three parts:

- The casting from float to rational on line 17 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 18 can cause an increase in the sensitivity equal to $n^{1/p} \cdot (2^k - 2^{k_{min}})$.

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

  This increase in the sensitivity is reflected on line 25, which, by the postcondition of `get_rounding_distance`, returns the maximum increase in sensitivity due to rounding, matching the above analysis.

- The discarding of the denominator on line 18 is 2^k-stable, as the denominator is 2^k. This increase in sensitivity is also reflected on line 25, where the sensitivity is multiplied by a factor of 2^{-k}.

For every pair of elements x, x' in `input_domain` and for every pair $(\texttt{d\_in}, \texttt{d\_out})$, 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`. □