fn make_int_to_bigint

Michael Shoemate

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

Proves soundness of the implementation of `make_int_to_bigint` in `mod.rs` at commit f5bb719 (outdated[1]).

1 Hoare Triple

Precondition

Compiler-Verified

- Generic `T` implements trait `SaturatingCast<IBig>`

User-Verified

None

Pseudocode

```
def make_int_to_bigint(
    input_space: tuple[VectorDomain[AtomDomain[T]], LpDistance[P, QI]],
) -> Transformation[
    VectorDomain[AtomDomain[T]],
    VectorDomain[AtomDomain[IBig]],
    LpDistance[P, QI],
    LpDistance[P, RBig],
]:
    input_domain, input_metric = input_space

    def stability_map(d_in):
        try:
            return RBig.try_from(d_in)
        except Exception:
            raise f"d_in ({d_in}) must be finite"

    return Transformation.new(
        input_domain,
        VectorDomain(
            element_domain=AtomDomain.default(IBig),
            size=input_domain.size,
        ),
        Function.new(lambda x: [IBig.from_(x_i) for x_i in x]),
        input_metric,
        LpDistance.default(),
        StabilityMap.new_fallible(stability_map),
    )
```

[1] See new changes with `git diff f5bb719..c3c9a76 rust/src/measurements/noise/nature/integer/mod.rs`

Postcondition

Theorem 1.1.

Theorem 1.2. For every setting of the input parameters (`T`) to `make_int_to_bigint` such that the given preconditions hold, `make_int_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. By the definition of the function on line **??**, and since `IBig.from` is infallible, the function is infallible, meaning that 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 **??**. Finally, the function is 1-stable, because the real values of the numbers remain un-changed, meaning the distance between adjacent inputs always remains the same, satisfying the stability property. □