

# 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Proves soundness of the implementation of `make_float_to_bigint_threshold` in `mod.rs` at commit `f5bb719` (outdated<sup>1</sup>).

## 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

```
1 def make_float_to_bigint_threshold(  
2   input_space: tuple[  
3     MapDomain[AtomDomain[TK], MapDomain[TV]], LOPI[P, AbsoluteDistance[QI]]  
4   ],  
5   k: i32,  
6 ) -> Transformation[  
7   MapDomain[AtomDomain[TK], AtomDomain[TV]],  
8   MapDomain[AtomDomain[TK], AtomDomain[IBig]],  
9   LOPI[P, AbsoluteDistance[QI]],  
10  LOPI[P, AbsoluteDistance[RBig]],  
11 ]:  
12   input_domain, input_metric = input_space  
13   if input_domain.value_domain.nan():
```

<sup>1</sup>See new changes with `git diff f5bb719..66159cf7 rust/src/measurements/noise_threshold/nature/float/mod.rs`

```

14         raise "input_domain hashmap values may not contain NaN elements"
15
16     def value_function(val): #
17         try: #
18             val = RBig.try_from(val)
19         except Exception:
20             val = RBig.ZERO
21
22         return find_nearest_multiple_of_2k(val, k) #
23
24     def stability_map(d_in):
25         l0, lp, li = d_in
26         rounding_distance = get_rounding_distance(k, usize.from_(l0), P)
27
28         lp = RBig.try_from(lp)
29         lp = x_mul_2k(lp + rounding_distance, -k) #
30
31         li = RBig.try_from(li)
32         li = x_mul_2k(li + rounding_distance, -k) #
33         return l0, lp, li
34
35     return Transformation.new(
36         input_domain,
37         MapDomain( #
38             key_domain=input_domain.key_domain,
39             value_domain=AtomDomain.default(IBig),
40         ),
41         Function.new(lambda x: {k: value_function(v) for k, v in x.items()}),
42         input_metric,
43         LOPI.default(),
44         StabilityMap.new_fallible(stability_map),
45     )

```

## 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_threshold` 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) ≤ 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. 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 37.

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 22 can cause an increase in the sensitivity equal to  $2^k$ .

$$\max_{x \sim x'} d_{Lp}(f(x), f(x')) \quad (1)$$

$$= \max_{x \sim x'} |r_k(x) - r_k(x')|_p \quad (2)$$

$$\leq \max_{x \sim x'} |(x + 2^{k-1}) - (x' - 2^{k-1})|_p \quad (3)$$

$$\leq \max_{x \sim x'} |x - x'|_p + 2^k \quad (4)$$

$$= \max_{x \sim x'} d_{Lp}(x, x') + 2^k \quad (5)$$

$$= 1 \cdot \mathbf{d\_in} + 2^k \quad (6)$$

This increase in the sensitivity is reflected on lines ?? and ??, where the rounding distance is added to the  $L_p$  and  $L_\infty$  sensitivities.

- The discarding of the denominator on line 22 is  $2^k$ -stable, as the denominator is  $2^k$ . This increase in sensitivity is also reflected on lines ?? and ??, where the sensitivity is multiplied by  $2^k$ .

For every pair of elements  $x, x'$  in `input_domain` and for every pair  $(\mathbf{d\_in}, \mathbf{d\_out})$ , where  $\mathbf{d\_in}$  has the associated type for `input_metric` and  $\mathbf{d\_out}$  has the associated type for `output_metric`, if  $x, x'$  are  $\mathbf{d\_in}$ -close under `input_metric`, `stability_map(d_in)` does not raise an exception, and `stability_map(d_in) ≤ d_out`, then `function(x), function(x')` are  $\mathbf{d\_out}$ -close under `output_metric`.  $\square$