

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Proves soundness of the implementation of `make_int_to_bigint` in `mod.rs` at commit `f5bb719` (outdated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

## 1 Hoare Triple

### Precondition

#### Compiler-Verified

- Generic T implements trait `SaturatingCast<IBig>`

#### User-Verified

None

### Pseudocode

```
1 def make_int_to_bigint(  
2     input_space: tuple[VectorDomain[AtomDomain[T]], LpDistance[P, QI]],  
3 ) -> Transformation[  
4     VectorDomain[AtomDomain[T]],  
5     VectorDomain[AtomDomain[IBig]],  
6     LpDistance[P, QI],  
7     LpDistance[P, RBig],  
8 ]:  
9     input_domain, input_metric = input_space  
10  
11     def stability_map(d_in):  
12         try:  
13             return RBig.try_from(d_in)  
14         except Exception:  
15             raise f"d_in ({d_in}) must be finite"  
16  
17     return Transformation.new(  
18         input_domain,  
19         VectorDomain(  
20             element_domain=AtomDomain.default(IBig),  
21             size=input_domain.size,  
22         ),  
23         Function.new(lambda x: [IBig.from_(x_i) for x_i in x]),  
24         input_metric,  
25         LpDistance.default(),  
26         StabilityMap.new_fallible(stability_map),  
27     )
```

<sup>1</sup>See new changes with `git diff f5bb719..f5c7b95 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.  $\square$