fn then_deintegerize_hashmap

Michael Shoemate

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

Proves soundness of the implementation of `then_deintegerize_hashmap` in `mod.rs` at commit f5bb719 (outdated[1]).

1 Hoare Triple

Precondition

Compiler-Verified

- Generic `TV` implements trait `CastInternalRational`

User-Verified

None

Pseudocode

```
def then_deintegerize_hashmap(k: i32) -> Function[HashMap[TK, IBig], HashMap[TK, TV]]:
    if k == i32.MIN: #
        raise ValueError("k must not be i32.MIN")

    def value_function(v_i):
        return TV.from_rational(x_mul_2k(RBig.from_(v_i), k))

    return Function.new(lambda x: {k_i: value_function(v_i) for k_i, v_i in x.items()})
```

Postcondition

Theorem 1.1. For every setting of the input parameters (`k, TK, TV`) to `then_deintegerize_hashmap` such that the given preconditions hold, `then_deintegerize_hashmap` raises an error (at compile time or run time) or returns a valid postprocessor. A valid postprocessor has the following property:

1. (Data-independent errors). For every pair of members x and x' in `input_domain`, `function`(x), `function`(x') either both raise the same error, or neither raise an error.

Proof. By the postcondition of `TV.from_rational`, the outcome of the function is the nearest representable float, and may saturate to positive or negative infinity. The precondition of `x_mul_2k` that k is not `i32.MIN` is satisfied on line 2. Since `TV.from_rational` and `x_mul_2k` are both infallible, the function is infallible, meaning that the function cannot raise data-dependent errors. Therefore the function is a valid postprocessor. □

[1] See new changes with `git diff f5bb719..6761ced rust/src/measurements/noise_threshold/nature/float/mod.rs`