fn then_saturating_cast_hashmap

Michael Shoemate

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

Proves soundness of the implementation of `then_saturating_cast_hashmap` in `mod.rs` at commit f5bb719 (outdated[1]).

1 Hoare Triple

Precondition

Compiler-Verified

- Generic `TK` implements trait `Hashable`
- Generic `TV` implements trait `SaturatingCast<IBig>`

User-Verified

None

Pseudocode

```
def then_saturating_cast_hashmap() -> Function[HashMap[TK, IBig], HashMap[TK, TV]]:
    return Function.new(lambda x: {k: TV.saturating_cast(v) for k, v in x.items()})
```

Postcondition

Theorem 1.1. For every setting of the input parameters (`TK, TV`) to `then_saturating_cast_hashmap` such that the given preconditions hold, `then_saturating_cast_hashmap` raises an exception (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 elements x, x' in `input_domain`, `function`(x), `function`(x') either neither or both raise an error. If both raise an error, then they both raise the same error.

Proof. Since `T.saturating_cast` is 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..8cbc2c7f rust/src/measurements/noise_threshold/nature/integer/mod.rs`