# fn then_saturating_cast

Michael Shoemate

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

Proves soundness of the implementation of `then_saturating_cast` in `mod.rs` at commit f5bb719 (outdated[1]).

## 1 Hoare Triple

### Precondition

#### Compiler-Verified

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

#### User-Verified

None

### Pseudocode

```
def then_saturating_cast() -> Function[Vec[IBig], Vec[TO]]:
    return Function.new(lambda x: [TO.saturating_cast(x_i) for x_i in x])
```

### Postcondition

**Theorem 1.1.** For every setting of the input parameters (`TO`) to `then_saturating_cast` such that the given preconditions hold, `then_saturating_cast` 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 `TO.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..11f4a00a rust/src/measurements/noise/nature/integer/mod.rs`