fn find_nearest_multiple_of_2k

Michael Shoemate

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

Proves soundness of the implementation of `find_nearest_multiple_of_2k` in `mod.rs` at commit f5bb719 (outdated[1]).

1 Hoare Triple

Precondition

Compiler-Verified

None

User-Verified

None

Pseudocode

```
def find_nearest_multiple_of_2k(x: RBig, k: i32) -> IBig:
    # exactly compute x/2^k and break into fractional parts
    num, den = x_mul_2k(x, -k).into_parts() #

    # argmin_i |i * 2^k - x|, the index of nearest multiple of 2^k
    return (floor_div(num << 1, den) + 1) >> 1 #
```

Postcondition

Theorem 1.1. Return $\max \text{argmin}_i |i2^k - x|$.

Proof. We first rewrite the expression into an efficiently computable integer form.

$$
\begin{aligned}
& \max \text{argmin}_i |i2^k - x| && && (1)\\
&= \max \text{argmin}_i |i - x2^{-k}| && \text{argmin doesn't change when rescaled} && (2)\\
&= \lfloor x2^{-k} \rceil && \text{where } \lfloor \cdot \rceil \text{ rounds to nearest int with ties towards infinity} && (3)\\
&= \lfloor n/d \rceil && \text{where } x2^{-k} = n/d, n \in \mathbb{Z}, n \in \mathbb{Z}^+ && (4)\\
&= ((n \cdot 2)//d + 1)//2 && \text{where } // \text{ denotes integer floor division} && (5)\\
&= ((n << 1)//d + 1) >> 1 && \text{where « and » denote left and right shift} && (6)
\end{aligned}
$$

Since x is a rational, line 3 splits $x2^{-k}$ into its numerator n and denominator d. Then line 6 directly computes the quantity of interest. The formula works by dividing twice the numerator, which retains whether

[1] See new changes with `git diff f5bb719..92ee78b9 rust/src/measurements/noise/nature/float/utilities/mod.rs`

the original number was in the lower or upper half of the interval. Addition by one moves the upper half of each interval to the next interval. Finally, floor division by two maps the data into the expected space.

Several implementation considerations are worth noting:

- The left shift operator `<<` is equivalent to integer multiplication by 2.
- The right shift operator `>>` is equivalent to floor integer division by 2, as the right-most bit is discarded, which always floors in two's complement arithmetic.
- Integer floor division is implemented via `floor_div`, because IBig integer division rounds towards zero, giving the wrong result on negative numerators.

Therefore the implementation follows the formula, which is equivalent to the postcondition. □