fn score_candidates_constants

Michael Shoemate

April 16, 2025

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

Proves soundness of `score_candidates_constants` in `mod.rs` at commit f5bb719 (outdated[1]). `score_candidates_consta` returns a fractional decomposition of alpha and an upper bound on dataset size to protect against overflow.

1 Hoare Triple

Precondition

None

Function

```python
def score_candidates_constants(size: Optional[u64], alpha: f64) -> tuple[u64, u64, u64]:
    if alpha < 0.0 or 1.0 < alpha:
        return ValueError("alpha must be within [0, 1]")

    alpha_num_exact, alpha_den_exact = RBig.try_from(alpha).into_parts()

    if size is not None:
        # choose the finest granularity that won't overflow
        # must have that size * denom < MAX, so let denom = MAX // size
        alpha_den_approx = u64.MAX.neg_inf_div(size)
    else:
        # default to an alpha granularity of .00001
        u64.exact_int_cast(10_000)

    if alpha_den_exact < UBig.from_(alpha_den_approx):
        alpha_num = u64.try_from(alpha_num_exact.into_parts()[1])
        alpha_den = u64.try_from(alpha_den_exact)
    else:
        # numer = alpha * denom
        alpha_num_approx = u64.round_cast(alpha * f64.round_cast(alpha_den_approx))
        alpha_num, alpha_den = alpha_num_approx, alpha_den_approx

    if size is not None:
        size_limit = size
    else:
        size_limit = u64.MAX.neg_inf_div(alpha_den)

    assert alpha_num <= alpha_den  #
    size_limit.alerting_mul(alpha_den)  #

    return alpha_num, alpha_den, size_limit  #
```

[1] See new changes with `git diff f5bb719..4ec9c684 rust/src/transformations/quantile_score_candidates/mod.rs`

Postcondition

Theorem 1.1. When the function does not fail, it returns the following constants:

- `alpha_num`
- `alpha_den`
- `size_limit`

The constants follow the following properties:

1. $alpha_num/alpha_den \in [0, 1]$
2. $size_limit \cdot alpha_den < 2^{64}$

An error is raised if these properties cannot be met.

Proof. While the code is complicated, all that is needed is to observe that the function does not return until line 31, and that the conditions are explicitly checked on lines 28 and 29.

Therefore, the postcondition holds. □