`impl InverseCDF for CanonicalRV`

Aishwarya Ramasethu, Yu-Ju Ku, Jordan Awan, Michael Shoemate

May 17, 2025

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

Proves soundness of the implementation of `InverseCDF` for `CanonicalRV`.
The implementation computes the inverse CDF of the canonical noise random variable.

1 Hoare Triple

Preconditions

Compiler-verified

- Argument `self` of type `CanonicalRV`.
- Argument `uniform` of type `RBig`.
- Argument `refinements` of type `usize`.
- Generic `R` implements `ODPRound` which denotes the rounding mode, either up or down.

Human-verified

Argument `uniform` is in $[0, 1]$.

Pseudocode

```
class InverseCDF(CanonicalRV):
    # type Edge = RBig

    def inverse_cdf(self, uniform: RBig, _refinements: usize, _R) -> RBig | None:
        f_inv = quantile_cnd(uniform, self.tradeoff, self.fixed_point)  #
        return f_inv * self.scale + self.shift
```

Postcondition

Theorem 1.1. Given a random variable `self` (of type `CanonicalRV`), the algorithm returns `Some(out)` where `out` is the inverse cumulative distribution function of `CanonicalRV` (which includes a rescale and shift) evaluated at `uniform` with error in direction `R`, or `None`.

The error between `out` and the exactly-computed CDF decreases monotonically as `refinements` increases.

Proof. By the definition of `CanonicalRV`,

- `self.tradeoff` is a symmetric nontrivial tradeoff function
- `self.fixed_point` is the fixed point of `self.tradeoff`, where `tradeoff(fixed_point) = fixed_point`.

Therefore the preconditions of `quantile_cnd` are met, so `f_inv` on line 5 is a sample from the canonical noise distribution with shift of zero and scale of one.

The function then returns the outcome, scaled by `self.scale` and shifted by `self.shift`.

Computing F_f^{-1}, rescaling and shifting are exact via fractional arithmetic, so making use of refinements is not necessary, as the error in computing the inverse CDF is already zero, satisfying the monotonicity property of the error.

$\square$