fn sample_bernoulli_exp1

Michael Shoemate

October 2, 2025

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

Proves soundness of `fn sample_bernoulli_exp1` in `mod.rs` at commit 0be3ab3e6 (outdated[1]). `fn sample_bernoulli_exp1` returns a sample from the $Bernoulli(exp(-x))$ distribution for some rational argument in $[0, 1]$. This proof is an adaptation of subsection 5.1 of [CKS20].

Vetting History

- Pull Request #519

1 Hoare Triple

Preconditions

- `x` is of type `Rational` and $x \in [0, 1]$
- `SampleBernoulli` is implemented for `Rational` probabilities

Pseudocode

```
def sample_bernoulli_exp1(x) -> bool:
    k = 1
    while True:
        if sample_bernoulli_rational(x / k, false): #
            k += 1
        else:
            return is_odd(k) #
```

Postcondition

For any setting of the input parameter `x` such that the given preconditions hold, `sample_bernoulli_exp1` either returns `Err(e)` due to a lack of system entropy, or `Ok(out)`, where `out` is distributed as $Bernoulli(exp(-x))$.

2 Proof

Assume the preconditions are met.

Lemma 2.1. `sample_bernoulli_exp1` only returns `Err(e)` when there is a lack of system entropy.

[1]See new changes with `git diff 0be3ab3e6..bf9b9f8 rust/src/traits/samplers/cks20/mod.rs`

Proof. In all usages of SampleBernoulli, the argument passed satisfies its definition preconditions, by the preconditions on `x` and function logic. Thus, by its definition, `sample_bernoulli_rational` only returns an error when there is a lack of system entropy. The only source of errors in `sample_bernoulli_exp1` is from the invocation of `sample_bernoulli_rational`. Therefore `sample_bernoulli_exp1` only returns `Err(e)` when there is a lack of system entropy. □

Lemma 2.2. Let k^* denote the final value of `k` on line 7. Then $P[K^* > n] = \frac{x^n}{n!}$ for any integer $n > 0$ [CKS20].

Proof. For $i \geq 0$, let a_i denote the i^{th} outcome of `sample_bernoulli_rational` on line 4. By the definition of `sample_bernoulli_rational`, under the established conditions and preconditions, each A_i is distributed as $Bernoulli(x/i)$.

$$\begin{aligned} P[K^* > n] &= P[A_1 = A_2 = ... = A_n = \top] && \text{since } K^* > n, \forall i \leq n, a_i = \top \\ &= \prod_{k=1}^{n} P[A_k = \top] && \text{all } A_i \text{ are independent} \\ &= \prod_{k=1}^{n} \frac{x}{k} && \text{since } A_k \sim Bernoulli(x/k) \\ &= \frac{x^n}{n!} \end{aligned}$$

□

Lemma 2.3. *is_odd*$(K^*) \sim Bernoulli(exp(-x))$ [CKS20].

Proof.

$$\begin{aligned} P[K^* \text{ odd}] &= \sum_{k=0}^{\infty} P[K^* = 2k+1] \\ &= \sum_{k=0}^{\infty} (P[K^* > 2k] - P[K^* > 2k+1]) \\ &= \sum_{k=0}^{\infty} \left(\frac{x^{2k}}{(2k)!} - \frac{x^{2k+1}}{(2k+1)!} \right) && \text{by 2.2} \\ &= exp(-x) \end{aligned}$$

Since `k` is distributed according to K^*, then `out` is distributed as $Bernoulli(exp(-x))$. □

Proof. 1 holds by 2.1 and 2.3. □

References

[CKS20] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *CoRR*, abs/2004.00010, 2020.