fn sample_bernoulli_exp

Michael Shoemate

June 11, 2025

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

Proves soundness of `sample_bernoulli_exp` in `mod.rs` at commit 0be3ab3e6 (outdated[1]).

`fn sample_bernoulli_exp` returns a sample from the $Bernoulli(exp(-x))$ distribution for some rational, non-negative, finite `x`. This proof is an adaptation of subsection 5.1 of [CKS20].

Vetting history

- Pull Request #519

1 Hoare Triple

Precondition

$\texttt{x} \in \mathbb{Q} \wedge \texttt{x} > 0$

Pseudocode

```
def sample_bernoulli_exp(x) -> bool:
    while x >= 1:
        if sample_bernoulli_exp1(1):  #
            x -= 1
        else:
            return False
    return sample_bernoulli_exp1(x)   #
```

Postcondition

For any setting of the input parameters `x` such that the given preconditions hold, `sample_bernoulli_exp` 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_exp` only returns `Err(e)` when there is a lack of system entropy.

[1] See new changes with `git diff 0be3ab3e6..8b0d898 rust/src/traits/samplers/cks20/mod.rs`

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

Lemma 2.2. `out` is distributed as $Bernoulli(exp(-x))$.

Proof. For $0 \leq i \leq \lfloor x \rfloor$, let b_i denote the i^{th} outcome of `sample_bernoulli_exp1` on line 3. By the definition of `sample_bernoulli_exp1`, under the established conditions and preconditions, each B_i is distributed as $Bernoulli(exp(-1))$. Let c denote the outcome of `sample_bernoulli_exp1` on line 7. Similarly as before, C is distributed $Bernoulli(exp(-(x - \lfloor x \rfloor)))$.

$$
\begin{aligned}
P[\texttt{out} = \top] &= P[B_1 = B_2 = ... = B_{\lfloor x \rfloor} = C = \top] && \texttt{out} \text{ is only } \top \text{ if } \forall i,\ B_i = \top \text{ and } C = \top \\
&= \prod_{i=1}^{\lfloor x \rfloor} P[B_i = \top] P[C = \top] && \text{all } B_i \text{ and } C \text{ are independent} \\
&= exp(-1)^{\lfloor x \rfloor} exp(\lfloor x \rfloor - x) \\
&= exp(-x)
\end{aligned}
$$

Therefore, `out` is distributed as $Bernoulli(exp(-x))$. □

Proof. 1 holds by 2.1 and 2.2. □

References

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