

# fn sample\_bernoulli\_exp

Michael Shoemate

January 30, 2026

Proves soundness of `sample_bernoulli_exp` in `mod.rs` at commit 0be3ab3e6 (outdated<sup>1</sup>).

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

## 1 Hoare Triple

### Precondition

#### Compiler-verified

- Argument  $x$  is of type `RBig`, a bignum rational

User-verified  $x > 0$

### Pseudocode

```
1 def sample_bernoulli_exp(x) -> bool:
2     while x > 1:
3         if sample_bernoulli_exp1(1):  #
4             x -= 1
5         else:
6             return False
7     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.

*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.  $\square$

<sup>1</sup>See new changes with `git diff 0be3ab3e6..4c1cf0c rust/src/traits/samplers/cks20/mod.rs`

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

*Proof.* For  $1 \leq i \leq \lfloor x \rfloor$ , 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  $Bernoulli(\exp(-1))$ . Similarly as before,  $C$  is distributed  $Bernoulli(\exp(-(x - \lfloor x \rfloor)))$ .

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

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.