fn sample_uniform_uint_below

Michael Shoemate

August 22, 2025

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

This document proves that the implementation of `sample_uniform_uint_below` in `mod.rs` at commit f5bb719 (outdated[1]) satisfies its definition.

`sample_uniform_uint_below` uses rejection sampling. In each round all bits of the integer are filled randomly, drawing an unsigned integer uniformly at random. The algorithm returns the sample, modulo the upper bound, so long as the sample is not one of the final "div" largest integers.

PR History

- Pull Request #473

0.1 Hoare Triple

Preconditions

- **Arguments**
  - `upper` must be of type `T` and non-negative
- **Type Arguments**
  - `T` has traits `Integer + Unsigned + FromBytes<N>`, which narrows valid types to u16, u32, u64, u128, usize
  - `N`. By the definition of `FromBytes`, the compiler ensures this is the number of bytes in T.

Pseudocode

```
def sample_uniform_uint_below(upper: T) -> T:
    threshold = T.MAX - T.MAX % upper

    while True:
        sample = sample_from_uniform_bytes()
        if sample < threshold:
            return sample % upper
```

Postcondition

For any setting of the input parameter `upper`, `sample_uniform_uint_below` either

- raises an exception if there is a lack of system entropy,
- returns `out` where `out` is uniformly distributed between $[0, \mathit{upper})$.

[1] See new changes with `git diff f5bb719..01f93a0 rust/src/traits/samplers/uniform/mod.rs`

0.2 Proof

Proof. By the postcondition of `sample_from_uniform_bytes`, then `sample` is a sample between zero and `T.MAX` inclusive, the greatest representable number of type `T`.

You could sample one of `upper` values uniformly at random by rejecting `sample` if it is larger than `upper`. That is, only return `sample` if `sample` is less than `upper`.

It is equivalent to extend the acceptance region, by returning `sample % 2` if `sample` is less than `sample * 2`, so long as `sample * 2 <= T.MAX`. This reduces the rejection rate, which increases algorithm performance.

There are `T.MAX % upper` remaining elements if you were to extend the acceptance region to the greatest multiple of `upper` that is less than `T.MAX`. Therefore conditioning `sample` on being less than `T.MAX - T.MAX % upper` results in `sample % upper` being an unbiased, uniformly distributed sample.
Therefore, for any value of `upper`, the function satisfies the postcondition. □