fn sample_uniform_ubig_below

Michael Shoemate

July 27, 2025

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

This document proves that the implementation of `sample_uniform_ubig_below` in `mod.rs` at commit f5bb719 (outdated[1]) satisfies its definition. This algorithm uses the same algorithm and argument as used for unsigned native integers, but this time the bit depth is dynamically chosen to fill the last byte of a series of bytes long enough to hold `upper`.

PR History

- Pull Request #473

1 Hoare Triple

Preconditions

- **Arguments**
  - `upper` of type `UBig`

Pseudocode

```python
def sample_uniform_ubig_below(upper: UBig) -> UBig:
    byte_len = upper.bit_len().div_ceil(8)
    max = Ubig.from_be_bytes([u8.MAX] * byte_len)
    threshold = max - max % upper

    buffer = [0] * byte_len
    while True:
        fill_bytes(buffer)

        sample = UBig.from_be_bytes(buffer)
        if sample < threshold:
            return sample % upper
```

Postcondition

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

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

[1] See new changes with `git diff f5bb719..0444870 rust/src/traits/samplers/uniform/mod.rs`

2 Proof

Proof. `byte_len` is the fewest number of bytes necessary to represent `upper`, which works out to $\text{ceil}(\text{ceil}(\log_2(\texttt{upper}))/8)$. Let `max` denote the largest integer representable in this many bytes ($2^{\texttt{byte\_len}} - 1$). We can sample uniformly from $[0, \texttt{max})$ by filling this many bytes with bits uniformly at random.

You could (naively) sample from $[0, \texttt{upper})$ by rejecting any `sample` greater than or equal to `upper`. To reduce the probability of rejecting (and improve computational performance), partition the numbers into two sets:

- the leading $\texttt{upper} \cdot k = \texttt{threshold}$ numbers that wrap evenly modulo `upper`
- the remaining trailing $(\texttt{max} \mod \texttt{upper})$ numbers

It is equivalent to only reject trailing numbers, and return the sample modulo `upper`. Since $\texttt{max} = \texttt{threshold} + (\texttt{max} \mod \texttt{upper})$, then $\texttt{threshold} = \texttt{max} - (\texttt{max} \mod \texttt{upper})$.
Therefore, for any value of `upper`, the function satisfies the postcondition. □