trait SampleUniformIntBelow

Michael Shoemate

January 30, 2024

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

PR History

- Pull Request #473

This document proves that the implementations of `SampleUniformIntBelow` in `mod.rs` at commit f5bb719 (outdated[1]) satisfy the definition of the `SampleUniformIntBelow` trait.

Definition 0.1. The `SampleUniformIntBelow` trait defines a function `sample_uniform_int_below`. For any setting of the input parameter `upper`, `sample_uniform_int_below` either

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

There are two `impl`'s (implementations): one for unsigned integers, and one for big integers. To prove correctness of each `impl`, we prove correctness of the implementation of `sample_uniform_int_below`.

Contents

1 `impl` for Unsigned Integers

This corresponds to `impl SampleUniformIntBelow for $ty` in Rust. `sample_uniform_int_below` uses rejection sampling. In each round, an unsigned integer is drawn uniformly at random 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.

1.1 Hoare Triple

Preconditions

- **User-specified types:**
  - Variable `upper` must be of type `T`
  - Type `T` is the type the trait is implemented for (one of u8, u16, u32, u64, u128, usize)

[1] See new changes with `git diff f5bb719..a1117d02 rust/src/traits/samplers/uniform/mod.rs`

Pseudocode

```
# returns a single bit with some probability of success
def sample_uniform_int_below(upper, T) -> int:
    while True:
        v = T.sample_uniform_int():
        if v < T.MAX - T.MAX % upper:
            return v % upper
```

Postcondition

The postcondition is supplied by 0.1.

1.2 Proof

Proof. Assuming that `T.sample_uniform_int()` is correctly implemented, then `v` 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 `v` if it is larger than `upper`. That is, only return `v` if `v` is less than `upper`.

It is equivalent to extend the acceptance region, by returning `v % 2` if `v` is less than `upper * 2`, so long as `upper * 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 `v` on being less than `T.MAX - T.MAX % upper` results in `v % upper` being an unbiased, uniformly distributed sample.
Therefore, for any value of `upper`, the function satisfies the postcondition. □