# fn make\_select\_private\_candidate

## Michael Shoemate

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

Proves soundness of make\_select\_private\_candidate in mod.rs at commit f5bb719 (outdated<sup>1</sup>).

make\_select\_private\_candidate returns a Measurement that returns a release from measurement whose score is above threshold, or may fail and return nothing.

## 1 Hoare Triple

#### Precondition

### Compiler-verified

- Generic DI (input domain) is a type with trait Domain.
- Generic MI (input metric) is a type with trait Metric.
- MetricSpace is implemented for (DI, MI). Therefore MI is a valid metric on DI.
- Argument measurement is a measurement whose output metric is MaxDivergence, and releases a tuple (f64, T0), where T0 is some arbitrary type.
- Argument stop\_probability is of type f64.
- Argument threshold is of type f64.

#### **User-verified**

None

## Pseudocode

```
def make_select_private_candidate(
    measurement: Measurement,
    stop_probability: float,
    threshold: float,
) -> Measurement:
    if not 0 <= stop_probability < 1: #
        raise "stop_probability must be in [0, 1)"

if not threshold.is_finite():
        raise "threshold must be finite"

scale = None
    if stop_probability > 0.0:
```

<sup>&</sup>lt;sup>1</sup>See new changes with git diff f5bb719..a58e34b rust/src/combinators/select\_private\_candidate/mod.rs

```
ln_cp = (1.0).neg_inf_sub(stop_probability).inf_ln()
14
15
           scale = ln_cp.recip().neg().into_rational() #
16
17
      def function(arg):
           remaining_iterations = None
18
           if scale is not None:
19
               remaining_iterations = UBig.ONE + sample_geometric_exp_fast(scale) #
20
21
           while True:
               score, output = measurement(arg)
23
24
               if score >= threshold:
25
                   return score, output
26
27
               if remaining_iterations is not None:
28
                   remaining_iterations -= UBig.ONE
29
                   if remaining_iterations == UBig.ZERO:
30
                        return None
31
32
      return Measurement (
33
           input_domain=measurement.input_domain,
34
35
           input_metric=measurement.input_metric,
           output_measure=measurement.output_measure,
36
           function=function,
37
           privacy_map=lambda d_in: measurement.map(d_in).inf_mul(2),
38
```

#### Postcondition

For every setting of the input parameters (measurement, stop\_probability, threshold, DI, MI, TO) to make\_select\_private\_candidate such that the given preconditions hold, make\_select\_private\_candidate raises an exception (at compile time or run time) or returns a valid measurement. A valid measurement has the following properties:

- 1. (Data-independent runtime errors). For every pair of elements x, x' in input\_domain, function(x) returns an error if and only if function(x') returns an error.
- 2. (Privacy guarantee). For every pair of elements x, x' in input\_domain and for every pair  $(d_in, d_out)$ , where d\_in has the associated type for input\_metric and d\_out has the associated type for output\_measure, if x, x' are d\_in-close under input\_metric, privacy\_map(d\_in) does not raise an exception, and privacy\_map(d\_in)  $\leq$  d\_out, then function(x), function(x') are d\_out-close under output\_measure.

### 2 Proofs

This section shows that the pseudocode implements Theorem 3.1 in [1], where stop\_probability is  $\gamma$  and threshold is  $\tau$ .

**Theorem 2.1.** The function pseudocode is equivalent to Algorithm 1 in [1].

*Proof.* scale is  $-1/\ln(1-\gamma)$ , with arithmetic conservatively rounded down. The rounding down corresponds to higher effective stop probability. A higher effective stop probability results in a lower effective privacy usage  $\epsilon_0$ . This means the advertised privacy loss is a conservative overestimate.

Since  $\gamma$  is restricted to [0,1) by 6, scale is non-negative. Therefore in 20, the precondition of sample\_geometric\_exp\_fast is satisfied. In Algorithm 1, the mechanism is invoked once before potentially terminating, so one is added to the sample. remaining\_iterations, a sample from the shifted geometric distribution, is then equivalent to a count of the number of coin flips made until sampling one heads, as is used in Algorithm 1.

We then only run as many iterations as has been sampled, as in Algorithm 1. In the case where T is infinity and  $\gamma$  is zero, then the algorithm only terminates when score exceeds threshold.

If the measurement releases a score of NaN, then this is effectively treated as a score below negative infinity, as NaN will never compare greater than the threshold.

Otherwise, the rest of the algorithm is evidently equivalent.

## Theorem 2.2. make\_select\_private\_candidate is consistent with Theorem 3.1 of [1].

*Proof.* Since there is no limit on iterations,  $\epsilon_0$  is zero. By 2.1, the function is equivalent to Algorithm 1, so we can claim parts a-e of Theorem 3.1. Since measurement is a valid measurement, then we know that it satisfies  $\epsilon_1$ -DP. The privacy map returns  $2\epsilon_1 + \epsilon_0$  with arithmetic rounded conservatively up, which is consistent with part b in Theorem 3.1.

(Privacy guarantee.) By 2.2, assuming correctness of Theorem 3.1 part b in [1], then for every pair of elements  $x, x' \in \text{input\_domain}$  and every  $d_{MI}(x, x') \leq \text{d\_in}$  with  $\text{d\_in} \geq 0$ , if x, x' are  $\text{d\_in-close}$  then function(x), function(x') are privacy\_map(d\_in)-close under output\_measure (the Max-Divergence).

## References

[1] Jingcheng Liu and Kunal Talwar. Private selection from private candidates, 2018.