List of definitions used in the proofs

Sílvia Casacuberta, Grace Tian, and Connor Wagaman

June–July 2021

We use the following guideline: if a term appears in the preconditions & pseudocode section, then this term is defined in the "List of definitions used in the pseudocode" document. Otherwise, it appears in the "List of definitions used in the proofs" document.

In both cases, we maintain the terms in alphabetical order within each section. 'TODOs" should be included at the end of the corresponding section. On the other hand, "TODOs" which better specify an already-defined term should be included immediately following the definition of that term. Examples should never be part of the definition, but we encourage their use right after the definition of a term.

Contents

List of terms that have not yet been added

- ...

1 Mathematical operators

(silvia) Passed form the list of definitions used in the pseudocode doc. Merge with "Operations" section.

The notation we are using for various mathematical operations is, as of 24 June 2021, inspired by section 2 of `https://hal-ens-lyon.archives-ouvertes.fr/ensl-01529804/`

`file/crlibm.pdf`. (The notation may change in the future.) We plan to use a similar standardized notation for describing the semantics of MPFR.[1]

Definition 1.1 $(+, -, \times)$**.** The symbols $+$, $-$, and $\times$ represent the usual mathematical operations of addition, subtraction, and multiplication, respectively.

Definition 1.2 $(\oplus, \ominus, \otimes)$**.** The symbols $\oplus$, $\ominus$, and $\otimes$ denote the corresponding floating point operations in MPFR / in Rust arithmetic.

(connor) Maybe change "floating point operations in MPFR" to some other thing that it denotes. I'm not sure if they should "denote the corresponding floating point operations in MPFR", but I wanted to preserve the notation, and I wasn't sure what it should be defined on . . . .

(silvia) Use same notation in both pseudocode and proofs?

TODO Change to cast. (connor) I don't understand this todo; could someone else complete it?

Definition 1.3. Given a Rust value x, the function `val(x)` returns the actual (real or integer) number represented by the Rust value x.

1.1 Notes, todos, questions

TODO Add the cast definition as discussed on June 30.

TODO max, min, val(x) (give number represented by Rust value), see email from June 21. Subindex for type?

TODO Max and min require TotalOrd

TODO Add special float symbol for max, min? No, from discussion on June 28: as long as we pass as arguments things that have used float symbols, it's OK.

Question: Type in subindex vs casting type. (connor) What does "subindex for type" mean (see todo above for use of this phrase; meaning possibly relates to question in this line)? (silvia) I think it means deciding between, for example, $\texttt{max}_{\texttt{u32}}(x, y)$ or `cast(max(x, y), u32)`.

2 Data Representation

Definition 2.1 (Vector)**.** A vector v is an ordered list of objects.

Definition 2.2 (Set)**.** A set is an unordered list of objects.

Definition 2.3 (Multiset)**.** A multiset is a modification of the notion of a set which, unlike a set, allows for repetitions for each of its elements. The number of repetitions of an element in the multiset corresponds to the *multiplicity* of that element. **Question:** multiset vs MultiSet.

Remark 1. Given a vector v, we denote its multiset representation by $\mathrm{MultiSet}(v)$. This distinction is relevant because vectors are ordered objects whereas multisets are not. In the OpenDP library, all datasets are represented as vector domains, and therefore for any

[1]Library used in OpenDP to deal with floating point arithmetic.

vector v we need to use the notation MultiSet(v) when referring to its multiset representation to indicate that ordering should be dropped. **Question:** Clashes a bit with the MultiSet$(\mathcal{X})$ use in the PF; domain vs vector.

Example 1. Given MultiSet$(2, 3, 3, 5, 5, 5)$, element 2 has multiplicity 1, element 3 has multiplicity 2, and element 5 has multiplicity 3.

Question: Is this clear enough? Because this is very important. (connor) Note: As of July 12, 2021, it may be less important since we may not need multisets anymore – for example, we now have a "vector-first" definition for symmetric distance that we can use. (silvia) Still needs discussion: defining terms in a "vector-first" manner to avoid using MultiSet(v) in the proofs.

(silvia) Decide one of the following 2:

Definition 2.4 (Histogram notation, MultiSet version)**.** Let $h_x : \mathcal{X} \to \mathbb{N}$ be the histogram of a multiset $x \in$ MultiSet$(\mathcal{X})$ for some domain $\mathcal{X}$. That is, $h_x(z)$ denotes the number of occurrences of $z \in \mathcal{X}$ in multiset x (with multiplicities).

Definition 2.5 (Histogram notation, vector version)**.** For any vector v of elements of a domain `D`, h_v denotes the histogram of v. That is, for every element z of type `T`, $h_v(z)$ denotes the number of occurrences of $z \in$ `D` in the entries of vector v (with multiplicities).

3 Transformations & Stability relation

Definition 3.1 (Transformation)**.** A transformation T is a *deterministic* mapping from arbitrary data types of derived data values to arbitrary data types of derived data values. In Rust, a transformation is specified by the following attributes: input domain, output domain, function, input metric, output metric, and stability relation.

See "List of definitions used in the pseudocode" for further details of the pseudocode specification of a transformation.

Definition 3.2 (Stability parameter)**.** For some value of c, we say that a transformation T is *c-stable* if for all x, x' in the input domain $\mathcal{X}$, and for input metric $d_{\mathcal{X}}$ and output metric $d_{\mathcal{Y}}$,

$$d_{\mathcal{Y}}(T(x), T(x')) \leq c \cdot d_{\mathcal{X}}(x, x'). \tag{1}$$

We say that c is the *stability parameter* of T.

(silvia) Why does the PF define the stability relation like this? There are cases where the $c \cdot d_{\mathcal{X}}(x, x')$ multiplicative constant scheme does not quite work; e.g., for clamping with absolute distance the stability relation is $d_{out} \geq \min(d_{in}, U - L)$.

In the Rust code, the stability parameter c (which in the case of clamping is equal to 1) gets wrapped up inside of the stability relation property, and the end user can test it empirically. TODO Explain better – but perhaps in the general guidelines doc?

Definition 3.3 (Stability relation)**.** The *stability relation*, denoted Relation(d_{in}, d_{out}), is a boolean function which takes as input some d_{in}, d_{out} appropriately quantified and returns `True` if and only if the relation $d_{out} \geq c \cdot d_{in}$ for some specified value of c.

We can also write the stability relation in the following form:

$$\text{Relation}(d_{in}, d_{out}) = \begin{cases} \texttt{True} & \text{if } d_{out} \geq c \cdot d_{in} \\ \texttt{False} & \text{otherwise,} \end{cases} \tag{2}$$

specifying the concrete metrics $d_{\mathcal{X}}, d_{\mathcal{Y}}$ and types of d_{in}, d_{out} inside the function.

The associated type of $d_{\mathcal{X}}$ is equal to the type of d_{in}, and the associated type of $d_{\mathcal{Y}}$ is equal to the type of d_{out}. Importantly, such relations are sound but *not necessarily complete.* A transformation is considered *valid* if its stability relation is sound.

3.1 Stability for Randomness

The following lemma and corollary are used to prove the stability guarantee holds in random transformations like `make_impute`.

Lemma 1 (Stability for Randomness)**.** *We define a randomized function f : `DI` $\rightarrow$ `DO`.* Relation(d_{in}, d_{out}) = *True implies that for all $x, x' \in$ `DI` that are d_{in}-close, there exists a coupling (R, R') of the randomness of $f(x)$ and $f(x')$ such that for all $(r, r') \in$ Support(R, R'), $f_r(x)$ is d_{out}-close to $f_{r'}(x')$.*

(silvia) Proof?

The following corollary allows us to fix the random seed in random transformations to prove the stability guarantee.

Corollary 1. *For randomized function f : `DI` $\rightarrow$ `DO`,* Relation(d_{in}, d_{out}) = *True implies that for all $x, x' \in$ `DI` that are d_{in}-close and for all fixing of the randomness r of f (fixing seed of PRG), we have that $f_r(x)$ and $f_r(x')$ are d_{out}-close.*

3.2 Notes, todos, questions

TODO Add explanation on forward and backward map.

4 Metrics

(silvia) Very important to always specify the types and domains when presenting metrics! (remarks from 24/6 Prof. Vadhan's OH) So:

(silvia) From meeting on July 13: each metric definition must contain its associated type and the list of possible domains it works with. This is not explicitly specified in Rust (i.e., the metric is only a name), but every time we find a transformation / measurement which uses the metric, include the corresponding domain in the definition if it is not there already. It should also include what it means to be d_{in} close under that metric.

(connor) Because the Rust type on which a metric operates is covered in the pseudocode definitions, I think we just need to talk mathematically here, like "symmetric distance is defined on vectors".

The associated type of any input metric is the type of the corresponding d_{in}. In turn, the associated type of any output metric is the type of the corresponding d_{out}.

4.1 Symmetric distance

Definition 4.1 (Symmetric difference)**.** Given a vector domain `VectorDomain(D)`, the *symmetric difference* between any two vectors $u, v \in$ `VectorDomain(D)`, denoted $\mathrm{MultiSet}(u)$ $\Delta\mathrm{MultiSet}(v)$, corresponds to the multiset representation of elements which are in either u or v but not in their intersection.

(connor) I think we should have an explanation of what is means for an element to be in, for example, u but not in v. For example, if we have $u = \mathrm{MultiSet}(0, 0, 1)$ and $v = \mathrm{MultiSet}(0)$, is 0 in v or not? We want it to be that "the first 0 is in v, but the second 0 is not in v", but I don't think the current definition gets that message across.

TODO Improve this definition

(silvia) In all these metric definitions, be careful about associated type and not to say input/output type. We never think of metrics as functions in the usual sense; instead, the associated type of distance is the type of d_{in}. When defining the metrics, we need to specify to which type of domains it applies.

Example 2. Because a multiset can have repeated elements, for $a = \mathrm{MultiSet}(1, 2, 1)$ and $b = \mathrm{MultiSet}(1, 3)$, we have $a\Delta b = \mathrm{MultiSet}(1, 3, 2)$.

We introduce the notion of symmetric distance, which differs from symmetric difference in that it is the *cardinality* of the multiset instead of the multiset itself.

Definition 4.2 (Symmetric distance)**.** Given an arbitrary vector domain `VectorDomain(D)`, the *symmetric distance* between any two vectors $u, v \in$ `VectorDomain(D)`, which is denoted as $d_{Sym} = |\mathrm{MultiSet}(u)\Delta\mathrm{MultiSet}(v)|$, is equal to the cardinality of the symmetric difference between u and v. Symmetric difference is a metric defined on vector domains and has associated type `u32`.

(connor) Do you think we should leave out the associated type above? The associated type may be more of a job for the pseudocode definitions since types aren't really mentioned in this doc.

Definition 4.3 (d_{in}-close under d_{Sym})**.** For any two vectors $u, v \in$ `VectorDomain(D)` and d_{in} of type `u32`, we say that u, v are d_{in}-close under d_{Sym} whenever $d_{Sym}(u, v) = |\mathrm{MultiSet}(u)\Delta\mathrm{MultiSet}(v)| \leq d_{in}$.

The same definition holds for d_{out}.

Note: symmetric distance is the metric which is used in the OpenDP library, while the definition symmetric difference is only included for completeness in this document.

Because there is a bijection between histograms and multisets, we can also define the *symmetric distance* between u and v as the ℓ_1 distance between the histograms for u and v, denoted h_u and h_v (see Definition 2.4). Then, we equivalently obtain that the symmetric distance between u and v is

$$d_{\mathrm{Sym}}(u, v) = \|h_v - h_w\|_1 = \sum_{z \in \mathcal{X}} |h_u(z) - h_v(z)|.$$

The second equality follows from the definition of ℓ_1 distance.

(silvia) Add the above as claim?

(connor) Since these definitions are all written by us, I think we could say that we're defining symmetric distance as both things.

Alternative definition of symmetric distance.

Let u, v be vectors of elements drawn from domain $\mathcal{X}$. Define $m_v(\ell)$ as the multiplicity of element ℓ in vector v. For example, if v contains five instances of the number "21", then $m_v(21) = 5$.

An alternative definition of symmetric distance, then, is

$$d_{\text{Sym}}(u, v) = \sum_{z \in \mathcal{X}} |m_u(z) - m_v(z)|.$$

Question: Which definition should we go with? Original definition, alternative definition, or both?

Claim 1. Symmetric distance is a metric.

Note that null data values are still counted in the symmetric distance. Adding or removing null values still influences the count.

The path property of symmetric distance on sized domains.

Lemma 2 (Path property of d_{Sym} on `SizedDomain`)**.** *For any two vectors v, w of the same length (that is, for any two vectors $v, w \in$ `SizedDomain(D)` for some domain `D`), $d_{Sym}(v, w)$ is an even integer; i.e., $d_{Sym}(v, w) = 2k$ for some integer $k \geq 0$. Moreover, there exist k vectors $v = v_0, v_1, \ldots, v_k = w$ such that $d_{Sym}(v_i, v_{i+1}) = 2$ for all i.*

Note that the path property of symmetric distance only holds in bounded domains (i.e., `SizedDomain`); that is, when the length of the input vectors is fixed.

Lemma 3. *For any two vectors $v, w \in$ `SizedDomain(D)`, $d_{Sym} = 2$ if and only if we can change one element of v to obtain v' such that* $\text{MultiSet}(v') = \text{MultiSet}(w)$.

The above lemma follows directly from the definition of symmetric distance.

Lemma 4. *Given a function f on input domain `SizedDomain(D)`, if $d_{out}(f(v), f(w)) \leq c$ for all vectors v, w such that $d_{Sym}(v, w) = 2$, then f is $c/2$-stable.*

By the definition of c-stable (Definition 3.2), that f is $c/2$-stable is equivalent to stating that for all pairs v', w' in the input domain, the following holds:

$$d_{\mathcal{Y}}(f(v'), f(w')) \leq c/2 \cdot d_{Sym}(v', w').$$

Also equivalently, this means that the stability relation of f is

$$d_{out} \geq c/2 \cdot d_{in}.$$

4.2 Substitute distance

Definition 4.4 (Substitute distance)**.** We only define the *substitute distance* d_{Sym} on multisets with the same number of elements. On two multisets $u, v \in \text{MultiSet}(\mathcal{X})$ for some domain $\mathcal{X}$ where $|u| = |v|$, we say that the substitute distance $d_{Subs}(u, v)$ is equal to the cardinality of the relative complement $u \backslash v$, so $d_{Subs}(u, v) = |u \backslash v|$. (This can be thought of as fixing multiset u and finding how many elements in v are not represented in v.) (silvia) List of domains? Associated types? It is not in the library, so we cannot answer this – maybe we should drop it.

(connor) Substitute distance should have the same domain as symmetric distance since it is defined as half the symmetric distance. Probably the same "associated type", too, but I don't think we need to cover that here.

Alternatively, we can define d_{Subs} as

$$d_{Subs}(u, v) = \frac{1}{2} d_{\text{Sym}}(u, v).$$

Note that this metric is like a generalization of Hamming distance to multisets (recall that Hamming distance is defined on ordered objects). (silvia) If we say "recall" then Hamming distance should be defined and explained. TODO Add Hamming? It is never used, so I would drop it.

(silvia) I would not define d_{Subs} in terms of d_{Sym}; and instead write original definition and then add the 2-factor equality as a claim. (connor) I think they had defined d_{subs} in terms of d_{sym}.

Claim 2. Substitute distance is a metric.

Remark 2. As of June 24, substitute distance has been removed from the library as a possible dataset metric. Therefore, symmetric distance is currently the only available dataset metric in the OpenDP library.

4.3 Absolute distance

Definition 4.5 (Absolute distance)**.** Given two elements n, m in `AllDomain(T)`, where `T` denotes an arbitrary type with trait `Sub(Output=T)`, the absolute distance between n and m, denoted d_{Abs}, is defined as $d_{Abs}(n, m) = |n - m|$, where the horizontal bars represent absolute value. Absolute distance has associated generic type `Q`.

TODO What data types is this operation defined on? (silvia) I think any type `T` with trait `Sub(Output=T)`.

Definition 4.6 (d_{in}-close under d_{Abs})**.** For any two elements n, m in `AllDomain(T)`, where `T` denotes an arbitrary type with trait `Sub(Output=T)`, and d_{in} of generic type `Q`, we say that n, m are d_{in}-close under d_{Abs} whenever $d_{Abs}(n, m) = |n - m| \leq d_{in}$.

The same definition holds for d_{out}.

Question: Should we add (or move) this to the "List of definitions used in the pseudocode" doc? (silvia) If we do it, we should move all of the metrics. I like it more here, but that is also from when we were using preconditions, because then the metric names never appear in the pseudocode.

Claim 3. Absolute distance is a metric.

Symmetric distance does not have an associated generic type; it is hard-coded to `u32`. It is the only such metric in the library that is hard-coded to an specific type. For example, absolute distance has the associated generic type `Q`, as stated in Definition 4.5

This is why we write `metric = SymmetricDistance()`, which implicitly implies that the associated d_{in}, d_{out} has type `u32`, but `metric = AbsoluteValue(T)`, which implies that the associated d_{in}, d_{out} has type `T`. Therefore, per our pseudocode structure, all the type signatures follow directly from the domain and metrics specified.

4.4 Distances

Definition 4.7 (k-close)**.** For any metric d, we say that two elements u, v are k-close under d if $d(u, v) \leq k$. (silvia) Needs fixing after July 13 meeting: d_{in}, d_{out} is more general than metrics; e.g., (ϵ, δ)-DP.

For example, in the case of symmetric distance, vectors u and v are k-close whenever $d_{Sym}(u, v) = |\text{MultiSets}(v) \Delta \text{MultiSets}(u)| \leq k$. We remark that the type of k must correspond to the associated type of d.

We remark that the notion of k-closeness can be defined more generally without relying on metrics and instead only using d_{in}, d_{out}. If that is the case, it will be defined accordingly.

4.5 Notes, todos, questions

TODO Make sure to cite Programming Framework when required.