1 Introduction

Differential privacy is a framework for privacy that gives rigorous guarantees on the amount of data leakage any one person's data can be subjected to when releasing statistical data. Since being introduced in 2006 [7], differential privacy has become the gold standard for private statistical analysis. Differentially private algorithms, whose efficacy are characterized by a "privacy cost" ε, primarily rely on the addition of statistical noise, ensuring that statistical results remain approximately correct while preventing any one person's information from being revealed.

Differentially private algorithms are notoriously tricky to analyze for correctness; most famously, the Sparse Vector Technique (SVT) algorithm has gone through multiple iterations, some of which were later shown to completely fail at protecting privacy[10]. Previous implementations of differential privacy by Apple have similarly been shown to have an increase from the claimed privacy cost by a factor of up to 16 [11].

Thus, much work has been done on developing methods for automatic verification of differentially private algorithms, both in their overall privacy and in the specific privacy costs they claim to achieve. Because even for limited programs the problem of determining if a program is differentially private is undecidable[2], previous work tends to focus on semi-decidability or further restricting program models.

Recently, a line of work has emerged around **approximate liftings** [3, 5, 4, 9]. Approximate liftings are a generalization of probabilistic couplings, themselves a well-known technique in probability theory for analyzing relationships between random variables. Approximate liftings allow for a more structured proof approach to many algorithms that themselves are not conducive to a standard compositional analysis, such as SVT. Because of their structure, liftings also lend themselves to automated proof construction [1].

We first rewrite the major results of approximate liftings in {not program logic}[1]. We then use approximate liftings to demonstrate that a certain limited class of programs, first described in [6], are differentially private; interestingly, we show that our class of liftings completely characterizes this class of programs. Additionally, we demonstrate that the privacy of a natural generalization of this class of programs can be proven using liftings and almost immediately follows from the privacy of the smaller class.

2 Differential Privacy

Differential privacy is a mathematically robust approach to privacy; most generally, differential privacy ensures that it is unlikely for an adversary to distinguish between whether or not one person's data was used in a private algorithm. To do this, differentially private algorithms rely on randomization, especially through the addition of statistical noise.

More precisely then, for a fixed output σ of a private algorithm A, the probability of obtaining σ for a dataset with some individual Alex is close (measured by a multiplicative factor) to the

[1] not sure how to describe this, also not sure if worth mentioning

probability of obtaining σ for the same dataset with Alex removed or Alex's data changed.

We will consider **datasets** $\mathcal{X} \in X^n$ of size n, where X is the set of all possible rows in the dataset; each person is represented by a single row.

We next define what it means for datasets to be "similar" to each other.

Definition 2.1. Two datasets $\mathcal{X} = (x_1, \ldots, x_n), \mathcal{X}' = (x'_1, \ldots, x'_n) \in X^n$ are **adjacent** (notated $\mathcal{X} \sim \mathcal{X}'$) if $|\{i : x_i \neq x'_i\}| \leq 1$[2].

We thus formalize privacy under this framework as follows.

Definition 2.2 (Pure Differential Privacy)**.** A randomized algorithm A is ε-differentially private if, for all pairs of **adjacent** datasets X and X' and all events $E \subseteq \texttt{im}(A)$,

$$\mathbb{P}[A(X) \in E] \leq e^{\varepsilon}\mathbb{P}[A(X') \in E]$$

An extremely useful property of differential privacy is that differentially private programs can be **sequentially composed** with a linear degradation in privacy:

Theorem 2.3 (Standard Composition)**.** *If A is ε_1-differentially private and, for all σ, $B(\sigma, \cdot)$ is ε_2-differentially private, then $B(A(X), X)$ is $\varepsilon_1 + \varepsilon_2$-differentially private.*

Composition therefore allows us to view privacy parameters ε as a "budget" for privacy-leaking operations in a program. Many[3] common differentially private algorithms are thus built out of well-known private components combined together, which also lend themselves to straightforward analyses.

2.1 Sensitivity and the Laplace Mechanism

Because we are typically interested in analyzing *functions* of our raw dataset (for example, the average age of a town), it is often useful to examine differential privacy through a similar model - instead of comparing two adjacent datasets $X \sim X'$, we compare **queries** $f(X)$ and $f(X')$. In this world, we care about the *sensitivity* of functions: how much a function *changes* when considering adjacent inputs.

Definition 2.4. The (ℓ_1-)sensitivity of a function $f : X \to \mathbb{R}$, often denoted Δf, is defined as $\Delta f = \max_{X \sim X'} ||f(X) - f(X')||_1$.

Given a function's sensitivity, we can easily make it differentially private through the use of the **Laplace Mechanism**.

Definition 2.5. The Laplace distribution $\texttt{Lap}(\mu, b)$ with mean μ and spread parameter b is the probability distribution with probability density function $f(x) = \frac{1}{2b}\exp(-\frac{|x-\mu|}{b})$. If $\mu = 0$, we will often abbreviate $\texttt{Lap}(0, b)$ as $\texttt{Lap}(b)$.

[2] A common variant is to define adjacency by the removal or addition of an entry, rather than changing one

[3] generic platitude - reword

The Laplace Mechanism, as expected, simply adds noise sampled from the Laplace distribution to a query result.

Theorem 2.6 (Theorem 3.6 [8])**.** *For a function f with sensitivity Δ, $A(X) = f(X) + \texttt{Lap}(\frac{\Delta}{\varepsilon})$ is ε-differentially private.*

We will consider the scenario where we are given a potentially infinite *sequence* of real-valued query functions $q_0, q_1, \ldots$, each with sensitivity at most Δ.

2.2 Deciding Privacy

Because designing differentially private algorithms can be quite tricky, we would like to be able to automatically (i.e. algorithmically) verify whether or not a given program is private, especially for algorithms whose privacy proofs do not rely primarily on composition. Ideally, beyond just determining whether a program is private or not, if a program is private, we'd like to find a good bound on the privacy cost for the program as well.

Unfortunately, even for relatively simple programs, just the basic problem is undecidable.

Theorem 2.7 ([2])**.** *The problem of determining whether a program from a certain class of algorithms with assignments, conditionals, and while loops is ε-differentially private is undecidable*[4]*.*

Thus, we will derive a decision procedure for a very specific class of potentially private programs; in particular, this class of programs lends itself to a straightforward analysis by **approximate liftings**, which we now introduce.

3 Couplings and Liftings

Probabilistic couplings are a common tool in probability theory; intuitively, couplings allow for the joint analysis of nominally unrelated probabilistic processes.

Definition 3.1. A coupling between two distributions A and B is a joint distribution C such that $\pi_1(C) = A$ and $\pi_2(C) = B$, where $\pi_1(C)$ and $\pi_2(C)$ are the first and second marginals of C, respectively.

In particular, couplings can be useful when analyzing the relation between two probablistic processes; couplings were first formulated by [check name] to analyze the behaviour of markov chains and have close connections to concepts such as total variation distance and stochastic domination.

As useful as standard couplings are, however, we must use more powerful machinery to properly reason about privacy.

Approximate liftings [4, 5, 9, 3] allow us to apply couplings to the realm of differential privacy.

[4] rephrase?

Definition 3.2. Let A_1, A_2 be two probability spaces[5]. We say a distribution μ_1 on A_1 and μ_2 on A_2 are related by the ε**-lifting** of the relation $\Psi \subseteq A_1 \times A_2$ (written $\mu_1 \Psi^{\#\varepsilon} \mu_2$) if there exist two **witness distributions** μ_L, μ_R on $A_1 \times A_2$ such that

1. $\pi_1(\mu_L) = \mu_1$ and $\pi_2(\mu_R) = \mu_2$
2. $\mathtt{supp}(\mu_L), \mathtt{supp}(\mu_R) \subseteq \Psi$
3. $\sup_{E \subseteq A_1 \times A_2} (\mathbb{P}_{x \leftarrow \mu_L}[x \in E] - e^{\varepsilon} \mathbb{P}_{x \leftarrow \mu_R}[x \in E]) \leq 0$

The similarities between the third condition and the definition of differential privacy should be clear. Indeed, there is a close connection between approximate liftings and differential privacy:

Theorem 3.3. *An algorithm $A(X)$ is ε-differentially private if and only if, for all adjacent input sequences $X \sim X'$, $A(X)(=)^{\#\varepsilon} A(X')$.*

If we are solely aiming to show that a program is private, we can instead work with the following relaxation:

Theorem 3.4. *If for all adjacent input sequences $X \sim X'$ and outputs σ of A, $A(X)\{(a,b) : a = \sigma \implies b = \sigma\}^{\#\varepsilon} A(X')$, then $A(X)$ is $\varepsilon-$differentially private.*

As expected, the foundational results of differential privacy can be restated in terms of liftings:

Proposition 3.5 (Laplace Mechanism for Liftings)**.** *If $X_1 \sim \mathtt{Lap}(\mu_1, \frac{1}{\varepsilon})$ and $X_2 \sim \mathtt{Lap}(\mu_2, \frac{1}{\varepsilon})$, then $X_1(=)^{\#\varepsilon|\mu_1 - \mu_2|} X_2$.*

Theorem 3.6 (Composition of Liftings)**.** *Let A_1, B_2, A_2, B_2 be distributions over S_1, T_1, S_2, T_2, respectively and let $R_1 \subseteq S_1 \times T_1$, $R_2 \subseteq S_2 \times T_2$ be relations. If $A_1 R_1^{\#\varepsilon_1} B_1$ and $A_1 R_1 B_1 \implies A_2 R_2^{\#\varepsilon_2} B_2$, then $A_2 R_2^{\#\varepsilon_1 + \varepsilon_2} B_2$.*

The structure of theorems 3.4 and 3.6 suggests the format that coupling proofs of privacy take: given two "runs" of an algorithm on adjacent inputs, construct many smaller liftings between program variables in each run and compose these liftings together to show that a final implicatory lifting between the outputs of the two runs exists.

3.1 Proving SVT with couplings

A classic algorithm that requires analysis beyond standard composition is Sparse Vector Technique (SVT). Given a possibly infinite stream of inputs and a threshold value, SVT will output if the queries are above or below the threshold (with noise on both the query and the threshold).

Unusually for differentially private algorithms, SVT can output a potentially unbounded number of "below threshold" queries before the first c "above threshold"s (or vice-versa), where c is some constant set by the user; when $c = 1$, SVT is frequently also referred to as "Above (or Below) Threshold". Potential applications include, for example, checking that

[5]may need to formally rewrite this at some point

a series of inputs is within an expected range or, appropriately, privately determining the non-zero elements of a sparse vector.

Because SVT allows for a potentially unbounded number of "below threshold" query outputs, its analysis requires a non-standard approach; a naive composition approach that assigns a fixed cost to outputting the result of each query will immediately result in unbounded privacy cost as well. Indeed, the analysis of SVT is notoriously difficult, with multiple published attempts at privacy proofs that were later shown to be incorrect[6].

However, re-analyzing SVT using approximate liftings can be relatively simple.

Algorithm 1 Sparse Vector Technique

Input: $\mathcal{X} \in X^n$, $T \in \mathbb{R}$, $Q = q_1, \ldots \in (X^n \to \mathbb{R})^*$ with sensitivity Δ, $c \in \mathbb{N}$.

$\varepsilon_1, \varepsilon_2 \leftarrow \frac{\varepsilon}{2}, \rho \leftarrow \texttt{Lap}(\frac{\Delta}{\varepsilon_1})$, $count \leftarrow 0$
for $q_i \in Q$ **do**
  $z \leftarrow \texttt{Lap}(\frac{2c\Delta}{\varepsilon_2})$
  if $q_i(\mathcal{X}) + z \geq T + \rho$ **then**
    output $\top$
    $count \leftarrow count + 1$
    if $count \geq c$ **then**
      break
    end if
  else
    output $\bot$
  end if
end for

Theorem 3.7. *Sparse Vector Technique is ε-differentially private.*

Proof. Consider two runs of SVT with adjacent inputs $\mathcal{X} \sim \mathcal{X}'$, respectively. We are aiming to show that $SVT(\mathcal{X}, T, Q, c)\{(a, b) : a = \sigma \implies b = \sigma\}^{\#\varepsilon} SVT(\mathcal{X}', T, Q, c)$ is a valid lifting.

Fix some output $\sigma \in \{\bot, \top\}^n$. Let $A = \{i : \sigma_i = \top\}$ be the indices of queries that are measured to be above the threshold. Note that $|A| = c$.

For every program variable x, let $x\langle 1\rangle$ and $x\langle 2\rangle$ represent the value of x in $SVT(\mathcal{X}, T, Q, c)$ and $SVT(\mathcal{X}', T, Q, c)$, respectively, so, for example, $q_i(\mathcal{X})\langle 1\rangle = q_i(\mathcal{X})$ and $q_i(\mathcal{X})\langle 2\rangle = q_i(\mathcal{X}')$.

Let $\tilde{T} = T + \rho$. Then $\tilde{T} \sim \texttt{Lap}(T, \frac{\Delta}{\varepsilon_1})$, so $\tilde{T}\langle 1\rangle + \Delta(=)^{\#\varepsilon_1}\tilde{T}\langle 2\rangle$.

Let $S_i = q_i(\mathcal{X}) + z_i$, so $S_i \sim \texttt{Lap}(q_i(\mathcal{X}), \frac{2c\Delta}{\varepsilon_2})$.

For all i such that $0 \leq i < n$, $i \notin A$, we construct the lifting $z_i\langle 1\rangle(=)^{\#0} z_i\langle 2\rangle$.

Then note that $\tilde{T}\langle 1\rangle + \Delta = \tilde{T}\langle 2\rangle \wedge z_i\langle 1\rangle = z_i\langle 2\rangle \implies (S_i\langle 1\rangle < \tilde{T}\langle 1\rangle \implies S_i\langle 2\rangle < \tilde{T}\langle 2\rangle)$.

[6] A textbook analysis of SVT, along with a discussion of bugged versions and incorrect privacy proofs, can be found at [10]

For all $i \in A$, create the lifting $z_i\langle 1\rangle (=)^{\#\frac{\varepsilon_2}{c}} z_i\langle 2\rangle - q_i(\mathcal{X}) + q_i(\mathcal{X}') - \Delta$, or equivalently, $S_i\langle 1\rangle + \Delta (=)^{\#\frac{\varepsilon_2}{c}} S_i\langle 2\rangle$. Note that this costs $\frac{\varepsilon_2}{c}$ since $|q_i(\mathcal{X}) - q_i(\mathcal{X}')| \leq \Delta$.

Then

$$\tilde{T}\langle 1\rangle + \Delta = \tilde{T}\langle 2\rangle \wedge S_i\langle 1\rangle + \Delta = S_i\langle 2\rangle \implies (S_i\langle 1\rangle \geq \tilde{T}\langle 1\rangle \implies S_i\langle 2\rangle \geq \tilde{T}\langle 2\rangle)$$

Thus, for all i, $SVT(\mathcal{X}, T, Q, c)_i = \sigma_i \implies SVT(\mathcal{X}', T, Q, c)_i = \sigma_i$, so $SVT(\mathcal{X}, T, Q, c)\{(a, b) : a = \sigma \implies b = \sigma\}^{\#\varepsilon_1+\varepsilon_2} SVT(\mathcal{X}', T, Q, c)$.

By Theorem 3.4, SVT is ε-differentially private. □

4 Automatically Proving Privacy using Couplings

We begin by building up a program model for SVT-style algorithms. There are three major components of SVT: taking in a threshold value and adding Laplace noise to it, taking in input and adding Laplace noise to it, and comparing the noisy threshold to the noisy input.

TODO: add non-input transitions (still don't understand why they're useful) - very trivial since the guards of a non-input transition have to be `true`

4.1 Individual Transitions

We will model programs as finite state automata, with each state of the automaton representing a possible program state. Under this paradigm, we begin with the simplest possible program: a single transition between two program states.

Definition 4.1 (Transitions)**.** Let $\mathcal{C} = \{\texttt{true}, \texttt{insample} < \texttt{x}, \texttt{insample} \geq \texttt{x}\}$ be a set of **transition guards**. Let Q be a set of states with each state $q \in Q$ having an associated real value for a variable `x`. Additionally, let $P(q) : Q \to \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0}$ be a function that associates each state state with two **noise parameters** $P(q) = (d_q, d'_q)$. Finally, let Γ be a finite alphabet of **output symbols**.

Then given two states $q, q' \in Q$, a **transition** from q to q' is defined as the tuple $t = (q, q', c, \sigma, \tau)$ where $c \in \mathcal{C}$, $\sigma \in \Gamma \cup \{\texttt{insample}, \texttt{insample}'\}$, and τ is a boolean value denoting whether or not the stored value of `x` will be updated. At all times, we will assume that there exists most one transition from $q \in Q$ to $q' \in Q$.

Definition 4.2 (Taking a transition)**.** Fix some $\varepsilon > 0$, which we will treat as a program parameter.

Consider some program state $q \in Q$ with noise parameters $P(q) = (d_q, d'_q)$. Let $z \sim \texttt{Lap}(0, \frac{1}{d_q\varepsilon})$ and $z' \sim \texttt{Lap}(0, \frac{1}{d'_q\varepsilon})$ be independent random noises sampled from Laplace distributions with spread parameters $\frac{1}{d_q\varepsilon}$ and $\frac{1}{d'_q\varepsilon}$, respectively.

For a state $q \in Q$ and input value $\texttt{in} \in \mathbb{R}$ read at q, let $\texttt{insample} = \texttt{in} + z$ and $\texttt{insample}' = \texttt{in} + z'$ be noisy versions of the input read at q.

Let $\sigma \in \Gamma \cup \{(\texttt{insample}, a, b), (\texttt{insample}', a, b)\}$ be a potential output of a transition. For measurability reasons, note that we must associate real-valued outputs with an interval (a, b). Intuitively, $\sigma = (\texttt{insample}, a, b)$ represents the possibility that `insample` was output with a value in the interval (a, b).

Then given a initial value $x_q \in \mathbb{R}$ for $\texttt{x}$, a possible transition $t = (q, q', c, \sigma, \tau)$ from q to another state $q' \in Q$ is **taken** if c is true and, if $\sigma \in \{(\texttt{insample}, a, b), (\texttt{insample}', a', b')\}$, t outputs `insample` or `insample'` as appropriate with a value in the interval (a, b) or (a', b'), respectively.

In general, we will assume that at all states q, x_q has been sampled from some Laplace distribution $\texttt{Lap}(\hat{\mu_q}, \frac{1}{\hat{d}_q \varepsilon})$, where $\hat{d}_q$ is the spread parameter for `insample` at some previous state.

4.2 Privacy

Observation 4.3 (Transition probabilities)**.** *Consider a single transition $t = (q, q', c, \sigma, \tau)$ from state $q \in Q$ to $q' \in Q$. Additionally, let $x_q \sim \texttt{Lap}(\hat{\mu_q}, \frac{1}{\hat{d}_q \varepsilon})$ be the initial stored value of $\texttt{x}$ at q.*

Let σ_0 be a potential output of t. In particular, if $\sigma \in \{\texttt{insample}, \texttt{insample}'\}$, then let $\sigma_0 = (\texttt{insample}, a, b)$ or $\sigma_0 = (\texttt{insample}', a', b')$ as appropriate. Otherwise, $\sigma_0 = \sigma \in \Gamma$.

*Then, given an input $\texttt{in}$ at q, the **probability** that t is taken with output σ_0 is clearly*

$$\mathbb{P}[x_q, t, \texttt{in}, \sigma_0] = \begin{cases} \mathbb{P}[c \text{ is satisfied}] & \sigma_0 = \sigma \\ \mathbb{P}[c \text{ is satisfied} \wedge \texttt{insample} \in (a, b)] & \sigma_0 = (\texttt{insample}, a, b) \\ \mathbb{P}[\texttt{insample}' \in (a', b')]\mathbb{P}[c \text{ is satisfied}] & \sigma_0 = (\texttt{insample}', a', b') \end{cases}$$

where $\texttt{Lap}_{\mu,s}(x)$ is the PDF of a Laplace distribution with mean μ and spread parameter s.

Note, in particular, that c being satisfied and $\texttt{insample} \in (a, b)$ are not independent events.

More precisely, if $(u, v), (u', v') \in \mathbb{R}_\infty$ is defined as follows:

$$(u, v) = \begin{cases} (-\infty, \infty) & c = \texttt{true} \wedge \sigma_0 \neq (\texttt{insample}, a, b) \\ (a, b) & c = \texttt{true} \wedge \sigma_0 = (\texttt{insample}, a, b) \\ (-\infty, \texttt{x}_0) & a = \texttt{insample} < x \wedge \sigma_0 \neq (\texttt{insample}, a, b) \\ (a, \min(\texttt{x}_0, b)) & c = \texttt{insample} < x \wedge \sigma_0 = (\texttt{insample}, a, b) \\ (\texttt{x}_0, \infty) & c = \texttt{insample} \geq x \wedge \sigma_0 \neq (\texttt{insample}, a, b) \\ (\max(\texttt{x}_0, a), b) & c = \texttt{insample} \geq x \wedge \sigma_0 = (\texttt{insample}, a, b) \end{cases}$$

$$(u', v') = (a', b'),$$

then, given an input $\texttt{in}$ at q, the probability that t is taken with output σ_0 can be computed to be

$$\mathbb{P}[x_q, t, \texttt{in}, \sigma_0] = \begin{cases} \int_u^v \texttt{Lap}_{\texttt{in}, \frac{1}{d_q \varepsilon}}(z) dz & \sigma_0 \neq (\texttt{insample}', a', b') \\ \int_{u'}^{v'} \texttt{Lap}_{\texttt{in}, \frac{1}{d'_q \varepsilon}}(z') dz' \int_u^v \texttt{Lap}_{\texttt{in}, \frac{1}{d_q \varepsilon}}(z) dz & \sigma_0 = (\texttt{insample}', a', b') \end{cases}$$

where $\texttt{Lap}_{\mu,s}(x)$ is the PDF of a Laplace distribution with mean μ and spread parameter s.

Recall that $\texttt{in}$, in reality, represents a **function** of some underlying dataset. This means that 'closeness' in this context is defined as follows:

Definition 4.4 (Adjacency)**.** Two inputs $\texttt{in} \sim_\Delta \texttt{in}'$ are Δ-adjacent if $|\texttt{in} - \texttt{in}'| \leq \Delta$. If Δ is not specified, we assume that $\Delta = 1$.

We can now define what it means to be **differentially private**.

Definition 4.5 ($d\varepsilon$-differential privacy for a transition)**.** Given two initial values x_q, x_q', a transition $t = (q, q', c, \sigma, \tau)$ is $d\varepsilon$**-differentially private** for some $d > 0$ if $\forall \varepsilon > 0$, for all adjacent inputs $\texttt{in} \sim \texttt{in}'$ and possible outputs $\sigma \in \Gamma \cup \{(\texttt{insample}, a, b), (\texttt{insample}', a', b')\}$, $\mathbb{P}[x_q, t, \texttt{in}, \sigma] \leq e^{d\varepsilon}\mathbb{P}[x_q', t, \texttt{in}', \sigma]$.

Recall that x_q and x_q' are treated as random variables drawn from a Laplace distribution.

Note that we slightly redefine ε-differential privacy as $d\varepsilon$-differential privacy, treating ε as a universal scaling parameter that can be fine-tuned by users for their own purposes. In particular, we argue that this definition is functionally equivalent[7], since if we are targeting ε^*-differential privacy, we can always take $\varepsilon = \frac{\varepsilon^*}{d}$.

4.2.1 Couplings

For every transition t between two states q and q', we can show that t is differentially private using a series of liftings.

Lemma 4.6. *Suppose that* $x_q\langle 1\rangle \sim \texttt{Lap}(\hat{\mu_q}\langle 1\rangle, \frac{1}{\hat{d}_q\varepsilon}), x_q\langle 2\rangle \sim \texttt{Lap}(\hat{\mu_q}\langle 2\rangle, \frac{1}{\hat{d}_q\varepsilon})$ *are two initial values of* $\boldsymbol{x}$ *at state* $q \in Q$.

Consider some transition $t = (q, q', c, \sigma, \tau)$ *from* q *to* $q' \in Q$. *Let* $P(q) = (d_q, d_q')$.

Let $\boldsymbol{in}\langle 1\rangle \sim \boldsymbol{in}\langle 2\rangle$ *be arbitrary and let* $\sigma\langle 1\rangle$, $\sigma\langle 2\rangle$ *be random variables representing possible outputs of* t *given inputs* $\boldsymbol{in}\langle 1\rangle$ *and* $\boldsymbol{in}\langle 2\rangle$, *respectively.*

Then $\forall \varepsilon > 0$ *and for all* $\gamma_x, \gamma_q, \gamma_q' \in [-1, 1]$ *that satisfy the constraints*

$$\begin{cases} \gamma_q \leq \gamma_x & c = \boldsymbol{insample} < \boldsymbol{x} \\ \gamma_q \geq \gamma_x & c = \boldsymbol{insample} \geq \boldsymbol{x} \\ \gamma_q = 0 & \sigma = \boldsymbol{insample} \\ \gamma_q' = 0 & \sigma = \boldsymbol{insample}' \end{cases},$$

the lifting $\sigma\langle 1\rangle\{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon}\sigma\langle 2\rangle$ *is valid for* $d = (|\hat{\mu_q}\langle 1\rangle - \hat{\mu_q}\langle 2\rangle + \gamma_x|)\hat{d_q} + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q'|)d_q'$, *and therefore* t *is* $d\varepsilon$*-differentially private.*

[7] [6] notes that it is not entirely clear how this differs from standard differential privacy, but that the known decidability result does not apply here - maybe something to investigate

Proof. Fix $\varepsilon > 0$.

We will analyze the behaviour of two different **runs** of t, one with input $\texttt{in}$ and one with input $\texttt{in}'$.

At a high level, for every Laplace-distributed variable, we will couple the value of the variable in one run with its value in the other **shifted** by some amount.

We differentiate between the values of variables in the first and second run by using angle brackets $\langle k \rangle$, so, for example, we will take $x_q\langle 1\rangle$ to be the value of $\texttt{x}$ at state q in the run of t with input $\texttt{in}\langle 1\rangle$ and $x_q\langle 2\rangle$ to be the value of $\texttt{x}$ in the run of t with input $\texttt{in}\langle 2\rangle$.

We thus want to create the lifting $\sigma\langle 1\rangle\{(a,b) : a = \sigma \implies b = \sigma\}\sigma\langle 2\rangle$. We must guarantee two things: that if the first transition is taken, then the second is also taken and that both runs output the same value σ when taking the transition. Note that if $c = \texttt{true}$, the first condition is trivially satisfied and when $\sigma \in \Gamma$, the second condition is trivially satisfied.

Recall that that x_q is sampled from a Laplace distribution, so $x_q\langle 1\rangle \sim \texttt{Lap}(\hat{\mu_q}\langle 1\rangle, \frac{1}{\hat{d}_q\varepsilon})$ and $x_q\langle 1\rangle \sim \texttt{Lap}(\hat{\mu_q}\langle 2\rangle, \frac{1}{\hat{d}_q\varepsilon})$.

Then we can first create the lifting $x_q\langle 1\rangle + \gamma_x (=)^{\#(|\hat{\mu_q}\langle 1\rangle - \hat{\mu_q}\langle 2\rangle + \gamma_x|)\hat{d}_q\varepsilon} x_q\langle 2\rangle$.

Additionally, create the lifting $z\langle 1\rangle (=)^{\#(|-\texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma_q|)d_q\varepsilon} z\langle 2\rangle - \texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma_q$.

This is equivalent to creating the lifting $\texttt{insample}\langle 1\rangle + \gamma_q (=)^{\#(|-\texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma_q|)d_q\varepsilon} \texttt{insample}\langle 2\rangle$.

Finally, create the lifting $z'\langle 1\rangle (=)^{\#(|-\texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma'_q|)d'_q\varepsilon} z'\langle 2\rangle - \texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma'_q$. As before, this is equivalent to creating the lifting $\texttt{insample}'\langle 1\rangle + \gamma'_q (=)^{\#(|-\texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma'_q|)d'_q\varepsilon} \texttt{insample}'\langle 2\rangle$.

If $c = \texttt{insample} < \texttt{x}$ and $\gamma_q \leq \gamma_x$, then

$$\begin{aligned}
\texttt{insample}\langle 1\rangle < x_q\langle 1\rangle &\implies \texttt{in}\langle 1\rangle + z\langle 1\rangle < x_q\langle 1\rangle \\
&\implies \texttt{in}\langle 1\rangle + z\langle 2\rangle - \texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma_q < x_q\langle 2\rangle - \gamma_x \\
&\implies \texttt{insample}\langle 2\rangle < x_q\langle 2\rangle
\end{aligned}$$

Similarly, if $c = \texttt{insample} \geq \texttt{x}$ and $\gamma_q \geq \gamma_x$, then

$$\begin{aligned}
\texttt{insample}\langle 1\rangle \geq x_q\langle 1\rangle &\implies \texttt{in}\langle 1\rangle + z\langle 1\rangle \geq x_q\langle 1\rangle \\
&\implies \texttt{in}\langle 1\rangle + z\langle 2\rangle - \texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma_q \geq x_q\langle 2\rangle - \gamma_x \\
&\implies \texttt{insample}\langle 2\rangle \geq x_q\langle 2\rangle
\end{aligned}$$

With these liftings, we have ensured that if the first run takes transition t, then the second run does as well.

Now, if t outputs $\texttt{insample}$ and $\gamma_q = 0$, then clearly we have that $\texttt{insample}\langle 1\rangle = \texttt{insample}\langle 2\rangle$, so $\sigma\langle 1\rangle = (\texttt{insample}, a, b) \implies \sigma\langle 2\rangle = (\texttt{insample}, a, b)$.

Similarly, if $\gamma'_q = 0$, we have that $\sigma\langle 1\rangle = (\texttt{insample}', a, b) \implies \sigma\langle 2\rangle = (\texttt{insample}', a, b)$.

Thus, if t outputs a real number in the interval (a, b) in the first run, it must also output a real number in the same interval in the second run.

Thus, given the constraints

$$\begin{cases} \gamma_q \leq \gamma_x & c = \texttt{insample} < \texttt{x} \\ \gamma_q \geq \gamma_x & c = \texttt{insample} \geq \texttt{x} \\ \gamma_q = 0 & \sigma = \texttt{insample} \\ \gamma'_q = 0 & \sigma = \texttt{insample}' \end{cases},$$

this is sufficient to create the lifting $\sigma\langle 1\rangle\{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon}\sigma\langle 2\rangle$, where the cost $d = (|\hat{\mu_q}\langle 1\rangle - \hat{\mu_q}\langle 2\rangle + \gamma_x|)\hat{d_q} + (|-\texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma_q|)d_q + (|-\texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma'_q|)d'_q$.

By an application of theorem 3.4, $\mathbb{P}[x_q\langle 1\rangle, t, \texttt{in}\langle 1\rangle, \sigma] \leq e^{d\varepsilon}\mathbb{P}[x_q\langle 2\rangle, t, \texttt{in}\langle 2\rangle, \sigma]$. Because $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$ are arbitrary adjacent inputs and σ is an arbitrary possible output of t, this implies that t is $d\varepsilon$-differentially private. □

We can thus think of couplings for a transition as being parameterized by γ_x, γ_q, and γ_q.

Definition 4.7 (Coupling strategies for a transition)**.** A **coupling strategy** for a transition $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$ is a tuple $C_i = (\gamma_x^{(i)}, \gamma_i, \gamma'_i) \in [-1, 1]^3$.

Definition 4.8 (Validity of a coupling strategy)**.** A coupling strategy $C_i = (\gamma_x^{(i)}, \gamma_i, \gamma'_i)$ for a transition t_i is **valid** if the constraints

$$\begin{cases} \gamma_i \leq \gamma_x^{(i)} & c_i = \texttt{insample} < \texttt{x} \\ \gamma_i \geq \gamma_x^{(i)} & c_i = \texttt{insample} \geq \texttt{x} \\ \gamma_i = 0 & \sigma_i = \texttt{insample} \\ \gamma'_i = 0 & \sigma_i = \texttt{insample}' \end{cases},$$

are all satisfied.

In particular, a valid coupling proof gives an upper bound on the privacy cost of any individual transition.

Proposition 4.9. *For a transition t_i, if there exists a valid coupling strategy $C_i = (\gamma_x^{(i)}, \gamma_i, \gamma'_i)$ given initial $\boldsymbol{x}$ values centred at $\hat{\mu_q}\langle 1\rangle$ and $\hat{\mu_q}\langle 2\rangle$, then t_i is $d\varepsilon$-differentially private for some*

$$d \leq \max_{\boldsymbol{in}\langle 1\rangle \sim \boldsymbol{in}\langle 2\rangle} (|\hat{\mu_q}\langle 1\rangle - \hat{\mu_q}\langle 2\rangle + \gamma_x^{(i)}|)\hat{d_q} + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_i|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma'_i|)d'_q.$$

Proof. Follows immediately from lemma 4.6. □

4.3 Multiple Transitions

Of course, in practice we would like to analyze the behaviour of programs with more than two program states.

We start with *executions*, comprised of a sequence of transitions.

Definition 4.10 (Program executions)**.** Consider some set of program states Q. A program **execution** is a sequence of transitions $t_0, t_1, \ldots, t_{n-1}$ such that for all $i \in 0 \ldots n-1$, $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$ for some c_i, σ_i, τ_i. We will often notate an execution ρ as $\rho = q_0 \to q_1 \to \ldots \to q_n$.

Additionally, if $\tau_0 = \texttt{true}$ and $c_0 = \texttt{true}$ (i.e. the first transition in any execution always assigns its noised input value into $\texttt{x}$), then ρ is a **complete execution**. We call this condition the **initialization** condition.

Definition 4.11 (Taking a execution)**.** Consider some complete execution $\rho = q_0 \to \ldots \to q_n$ and let $\texttt{x}_i$ for all $i \in 1 \ldots n$ be the value of $\texttt{x}$ at each program state q_i. Note that we can safely ignore $\texttt{x}_0$ because of the initialization condition. As before, for a program state $q_i \in Q$ with noise parameters $P(q_i) = (d_{q_i}, d'_{q_i})$, let $z_i \sim \texttt{Lap}(0, \frac{1}{d_{q_i}\varepsilon})$ and $z'_i \sim \texttt{Lap}(0, \frac{1}{d'_{q_i}\varepsilon})$. Further, given an input $\texttt{in} \in \mathbb{R}^n$ and possible output $\sigma \in (\Gamma \cup \{(\texttt{insample}, a, b), (\texttt{insample}', a, b)\})^n$, let $\texttt{insample}_i = \texttt{in}_i + z_i$ and $\texttt{insample}'_i = \texttt{in}_i + z'_i$. $\texttt{x}$ will be updated with $\texttt{insample}$ at a transition t_i if and only if $\tau_i = \texttt{true}$. In other words, $\texttt{x}_i = \begin{cases} \texttt{x}_{i-1} & \tau_{i-1} = \texttt{false} \\ \texttt{insample}_{i-1} & \tau_{i-1} = \texttt{true} \end{cases}$.

Then ρ is **taken** if, $\forall i \in 0 \ldots n-1$, c_i is satisfied for $\texttt{insample}_i$ and $\texttt{x}_i$ and, if t_i outputs a real value, it is of the form $(\texttt{insample}_i, a, b)$ or $(\texttt{insample}'_i, a', b')$, as appropriate.

4.3.1 Privacy

We can naturally extend the definition of probability on a single transition to an entire execution recursively.

Definition 4.12. Given a execution $\rho = q_0 \to q_1 \to \ldots \to q_n$, the **tail** of ρ is defined as $tail(\rho) = q_1 \to q_2 \to \ldots \to q_n$.

We may additionally use the notation $\rho_{i:j}$ to represent the subexecution $q_i \to q_{i+1} \to \ldots \to q_j$ of ρ. Using this notation, $tail(\rho) = \rho_{i:} = \rho_{i:n}$.

Definition 4.13 (execution probabilities)**.** Consider a execution $\rho = q_0 \to q_1 \to \ldots \to q_n$ of length n. Let $t_0 = (q_0, q_1, c_0, \sigma_0, \tau_0)$ between states q and q'. Additionally, let $\texttt{x}_0$ be the initial value of $\texttt{x}$ at q_0.

If $\sigma_0 \in \{\texttt{insample}, \texttt{insample}'\}$, then let $\sigma_0 = (\texttt{insample}, a, b)$ or $\sigma_0 = (\texttt{insample}', a', b')$ as appropriate.

Additionally, let $(u, v), (u', v') \in \mathbb{R}_\infty$ be defined as follows:

$$(u, v) = \begin{cases} (-\infty, \infty) & c_0 = \texttt{true} \wedge \sigma_0 \neq (\texttt{insample}, a, b) \\ (a, b) & c_0 = \texttt{true} \wedge \sigma_0 = (\texttt{insample}, a, b) \\ (-\infty, \texttt{x}_0) & c_0 = \texttt{insample} < x \wedge \sigma_0 \neq (\texttt{insample}, a, b) \\ (c, \min(\texttt{x}_0, d)) & c_0 = \texttt{insample} < x \wedge \sigma_0 = (\texttt{insample}, a, b) \\ (\texttt{x}_0, \infty) & c_0 = \texttt{insample} \geq x \wedge \sigma_0 \neq (\texttt{insample}, a, b) \\ (\max(\texttt{x}_0, c), d) & c_0 = \texttt{insample} \geq x \wedge \sigma_0 = (\texttt{insample}, a, b) \end{cases}$$
$$(u', v') = (a', b')$$

Then, given an initial $\texttt{x}$-value $\texttt{x}_0$, an input sequence $\texttt{in} \in \mathbb{R}^n$, and an output sequence $\sigma \in (\Gamma \cup \mathbb{R})^n$, the probability of ρ is defined as

$$\mathbb{P}[\texttt{x}_0, \rho, \texttt{in}, \sigma] = \begin{cases} 1 & n = 0 \\ \int_u^v \texttt{Lap}_{\texttt{in}_0, \frac{1}{d_0 \varepsilon}}(z) dz \mathbb{P}[\texttt{x}_0, \rho_{1:}, \texttt{in}_{1:}, \sigma_{1:}] & \sigma_0 \neq \texttt{insample}' \wedge \tau_0 = \texttt{false} \\ \int_u^v \texttt{Lap}_{\texttt{in}_0, \frac{1}{d_0 \varepsilon}}(z) \mathbb{P}[z, \rho_{1:}, \texttt{in}_{1:}, \sigma_{1:}] dz & \sigma_0 \neq \texttt{insample}' \wedge \tau_0 = \texttt{true} \\ \int_{u'}^{v'} \texttt{Lap}_{\texttt{in}_0, \frac{1}{d_0' \varepsilon}}(z') dz' \int_u^v \texttt{Lap}_{\texttt{in}_0, \frac{1}{d_0 \varepsilon}}(z) dz \mathbb{P}[\texttt{x}_0, \rho_{1:}, \texttt{in}_{1:}, \sigma_{1:}] & \sigma_0 = \texttt{insample}' \wedge \tau_0 = \texttt{false} \\ \int_{u'}^{v'} \texttt{Lap}_{\texttt{in}_0, \frac{1}{d_0' \varepsilon}}(z') dz' \int_u^v \texttt{Lap}_{\texttt{in}_0, \frac{1}{d_0 \varepsilon}}(z) \mathbb{P}[z, \rho_{1:}, \texttt{in}_{1:}, \sigma_{1:}] dz & \sigma_0 = \texttt{insample}' \wedge \tau_0 = \texttt{true} \end{cases}$$

where $\texttt{Lap}_{\mu, s}(x)$ is the PDF of a Laplace distribution with mean μ and spread parameter s.

For a complete execution ρ, note that the initial value of $\texttt{x}$ is irrelevant, so we will shorthand $\mathbb{P}[\texttt{x}_0, \rho, \texttt{in}, \sigma]$ to $\mathbb{P}[\rho, \texttt{in}, \sigma]$.

Additionally, because we now read in a *sequence* of real-valued inputs, we need to slightly modify our definition of adjacency.

Definition 4.14 (Adjacency for a sequence of inputs)**.** Two input sequences $\{\alpha_i\}_{i=1}^n, \{\beta_i\}_{i=1}^n$ of length n are Δ-adjacent (notated $\alpha \sim_\Delta \beta$) if, for all $i \in [1 \dots n]$, $|\alpha_i - \beta_i| \leq \Delta$.

If Δ is not specified, we assume that $\Delta = 1$.

Thus, we get the following definition of privacy:

Definition 4.15 ($d\varepsilon$-differential privacy for a execution)**.** A complete execution ρ of length n is $d\varepsilon$-differentially private for some $d > 0$ if $\forall \varepsilon > 0$, for all adjacent input sequences $\alpha \sim \beta$ of length n and all possible output sequences σ of length n, $\mathbb{P}[\rho, \alpha, \sigma] \leq e^{d\varepsilon} \mathbb{P}[\rho, \beta, \sigma]$.

4.3.2 Concatenating couplings

Let $\rho[\texttt{in}]$ be a random variable representing the output of ρ given input sequence $\texttt{in}$.

In order to show that a program execution ρ is differentially private, for all adjacent inputs $\alpha \sim \beta$ and all possible outputs σ, we want to create the coupling $\rho[\alpha]\{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon} \rho[\beta]$ for some $d > 0$.

Ideally, we would like to simply create couplings for each individual transition in ρ as before and compose them together to create this overall coupling. Indeed, this approach is almost sufficient; the constraints imposed upon shifts for a coupling for transition t_i depend solely on the shift at the most recent **assignment transition** in ρ (i.e. the most recent transition t_j such that $\tau_j = \texttt{true}$). The coupling shifts for *non-assignment transitions* can thus never impact each other.

Definition 4.16 (Assignment transitions)**.** Let $A_\rho = \{t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i) : \tau_i = \texttt{true}\}$ be the set of **assignment transitions** in a execution ρ.

For every transition t_i in ρ, let $t_{at(i)}$ be the most recent assignment transition in ρ; i.e., $at(i) = \max\{j < i : t_j \in A_\rho\}$. If such a j does not exist, we set $at(i) = -1$.

In particular, note that for transition t_i, $\gamma_x = \gamma_{at(i)}$, where γ_{-1} is the shift applied to the initial $\texttt{x}$-values $\texttt{x}_0\langle 1\rangle$ and $\texttt{x}_0\langle 2\rangle$.

Thus, for an individual transition t_i of ρ. From proposition 4.9, we have a family of valid coupling strategies $C_i(\gamma_{at(i)}, \gamma_i, \gamma_i')$.

We can merge these coupling strategies together to create a proof of privacy for the entire execution:

Lemma 4.17. *Let $\rho = q_0 \to \ldots \to q_n$ be a complete execution of length n. Let $\boldsymbol{in}\langle 1\rangle \sim \boldsymbol{in}\langle 2\rangle$ be arbitrary adjacent input sequences of length n. Additionally, fix some potential output σ of ρ of length n and let $\sigma\langle 1\rangle$, $\sigma\langle 2\rangle$ be random variables representing possible outputs of ρ given inputs $\boldsymbol{in}\langle 1\rangle$ and $\boldsymbol{in}\langle 2\rangle$, respectively. Additionally, for all q_i, let $P(q_i) = (d_i, d_i')$.*

Then $\forall \varepsilon > 0$ and for all $\{\gamma_i, \gamma_i'\}_{i=0}^{n-1}$ that, for all i, satisfy the constraints

$$\begin{cases} \gamma_i \le \gamma_{at(i)} & c_i = \boldsymbol{insample} < \boldsymbol{x} \\ \gamma_i \ge \gamma_{at(i)} & c_i = \boldsymbol{insample} \ge \boldsymbol{x} \\ \gamma_i = 0 & \sigma_i = \boldsymbol{insample} \\ \gamma_i' = 0 & \sigma_i = \boldsymbol{insample}' \end{cases},$$

the lifting $\sigma\langle 1\rangle\{(a,b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon}\sigma\langle 2\rangle$ is valid for $d = \sum_{i=0}^{n-1}(|-\boldsymbol{in}_i\langle 1\rangle + \boldsymbol{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\boldsymbol{in}_i\langle 1\rangle + \boldsymbol{in}_i\langle 2\rangle - \gamma_i'|)d_i'$, and therefore t is $d\varepsilon$-differentially private.

Proof. From the proof of lemma 4.6, we know that we can create the couplings $\texttt{insample}_i\langle 1\rangle + \gamma_i (=)^{\#(|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i\varepsilon} \texttt{insample}_i\langle 2\rangle$ and $\texttt{insample}_i'\langle 1\rangle + \gamma_i' (=)^{\#(|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i'|)d_i'\varepsilon} \texttt{insample}_i'\langle 2\rangle$ for all q_i in ρ.

Additionally, for some fixed q_i in ρ, if we have the coupling $\texttt{x}_i\langle 1\rangle + \gamma_x (=)^{\#(|\hat{\mu_i}\langle 1\rangle - \hat{\mu_i}\langle 2\rangle + \gamma_x|)\hat{d}_i\varepsilon} x_i\langle 2\rangle$, where $\texttt{x}_i\langle 1\rangle \sim \texttt{Lap}(\hat{\mu_i}\langle 1\rangle, \frac{1}{\hat{d}_i\varepsilon})$ and $\texttt{x}_i\langle 2\rangle \sim \texttt{Lap}(\hat{\mu_i}\langle 2\rangle, \frac{1}{\hat{d}_i\varepsilon})$, then subject to the constraints

$$\begin{cases} \gamma_i \le \gamma_x & c_i = \texttt{insample} < \texttt{x} \\ \gamma_i \ge \gamma_x & c_i = \texttt{insample} \ge \texttt{x} \\ \gamma_i = 0 & \sigma_i = \texttt{insample}_i \\ \gamma_i' = 0 & \sigma_i = \texttt{insample}_i' \end{cases},$$

the coupling $\sigma_i\langle 1\rangle\{(a,b) : a = \sigma_i \implies b = \sigma_i\}^{\#d\varepsilon}\sigma_i\langle 2\rangle$ is valid for some d.

Indeed, note that for all i, $\mathtt{x}_i = \mathtt{insample}_{at(i)}$ by definition. Thus, we have that $\mathtt{x}_i\langle 1\rangle + \gamma_x(=)^{\#(|-\mathtt{in}_{at(i)}\langle 1\rangle+\mathtt{in}_{at(i)}\langle 2\rangle+\gamma_{at(i)}|)d_{at(i)}\varepsilon} x_i\langle 2\rangle$, and we must satisfy the constraints

$$\begin{cases} \gamma_i \leq \gamma_{at(i)} & c_i = \mathtt{insample} < \mathtt{x} \\ \gamma_i \geq \gamma_{at(i)} & c_i = \mathtt{insample} \geq \mathtt{x} \\ \gamma_i = 0 & \sigma_i = \mathtt{insample}_i \\ \gamma_i' = 0 & \sigma_i = \mathtt{insample}_i' \end{cases}$$

for all i.

Thus, we can put all of these couplings together to show that the coupling $\sigma_i\langle 1\rangle\{(a,b) : a = \sigma_i \implies b = \sigma_i\}^{\#d\varepsilon}\sigma_i\langle 2\rangle$ is valid for some $d > 0$.

In particular, note that we have created at most one pair of couplings (for `insample` and `insample`) for each q_i. Thus, the total coupling cost associated with each q_i is at most $(|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i'|)d_i'$, which gives us an overall coupling cost of $d = \sum_{i=0}^{n-1}(|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i'|)d_i'$. □

Definition 4.18. For a complete execution ρ of length n and adjacent input sequences $\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$, a **coupling strategy** is two functions $\boldsymbol{\gamma}(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) : \mathbb{R}^n \times \mathbb{R}^n \to [-1,1]^n$ and $\boldsymbol{\gamma}'(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) : \mathbb{R}^n \times \mathbb{R}^n \to [-1,1]^n$ that produce shifts for each transition of ρ dependent on the input sequences.

If $\mathtt{in}\langle 1\rangle$ and $\mathtt{in}\langle 2\rangle$ are clear from context, we will often shorthand notating a coupling strategy as $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}'$.

Definition 4.19. For a complete execution ρ of length n, a coupling strategy $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is **valid** if $\forall \mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$, $\boldsymbol{\gamma}(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)$ and $\boldsymbol{\gamma}'(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)$ satisfy the constraints

$$\begin{cases} \gamma_i \leq \gamma_{at(i)} & c_i = \mathtt{insample} < \mathtt{x} \\ \gamma_i \geq \gamma_{at(i)} & c_i = \mathtt{insample} \geq \mathtt{x} \\ \gamma_i = 0 & \sigma_i = \mathtt{insample} \\ \gamma_i' = 0 & \sigma_i = \mathtt{insample}' \end{cases}.$$

4.3.3 Optimizing Privacy

Definition 4.20. For a complete execution ρ of length n, the **cost** of a coupling strategy $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is

$$cost(C_\rho) = \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} \sum_{i=0}^{n-1} (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i'|)d_i'.$$

Additionally, let G be the set of all valid coupling strategies $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ for ρ. Then the **coupling cost** of ρ is

$$cost(\rho) = \min_{(\boldsymbol{\gamma},\boldsymbol{\gamma}')\in G} cost((\boldsymbol{\gamma}, \boldsymbol{\gamma}')).$$

As before, the existence of a valid coupling strategy upper bounds the privacy cost of any execution.

Proposition 4.21. *If $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is valid, then ρ is $cost(C_\rho)\varepsilon$-differentially private.*

Proof. Follows immediately from lemma 4.17. □

Corollary 4.22. *Any complete execution ρ is $cost(\rho)\varepsilon$-differentially private. Further, for all complete executions ρ, $cost(\rho) < \infty$.*

Proof. The first claim follows immediately from definitions.

The second claim follows by considering a coupling strategy $(\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ where $\forall \texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle, \boldsymbol{\gamma} = \boldsymbol{\gamma}' = \mathbf{0}$. Note that $(\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is trivially valid. Since $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$, $cost(\rho) \leq cost(C_\rho(\mathbf{0})) \leq \sum_{i=0}^{n-1}(d_i + d_i')$, which is finite for all fixed ρ. □

Proposition 4.23. *The cost of a coupling strategy over a fixed execution ρ is maximized when, for every transition, the difference between the input values is* 1*. In other words,*

$$\begin{aligned}
&\max_{\boldsymbol{in}\langle 1\rangle \sim \boldsymbol{in}\langle 2\rangle} \sum_{i=0}^{n-1} (|-\boldsymbol{in}_i\langle 1\rangle + \boldsymbol{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\boldsymbol{in}_i\langle 1\rangle + \boldsymbol{in}_i\langle 2\rangle - \gamma_i'|)d_i' \\
&= \max_{\Delta \in \{-1,1\}} \sum_{i=0}^{n-1} (|\Delta - \gamma_i|)d_i + (|\Delta - \gamma_i'|)d_i'
\end{aligned}$$

Proof. (Vishnu) □

Note then, that the process of finding the optimal coupling cost of a execution ρ can be formulated as a linear program.

4.4 Branching

Definition 4.24 (Branching program)**.** Given a set of program states Q, a branching program B is a finite set of complete executions over Q such that for all $\rho = q_0 \to \ldots \to q_n, \rho' = q_0' \to \ldots \to q_m' \in B$, $q_0 = q_0'$. B must also satisfy the following properties:

1. If any transition t in any execution $\rho \in B$ is of the form (q, q', c, σ, τ), then no other transitions of the form $(q, q^*, c, \sigma', \tau')$ for $q, q', q^* \in Q$ exist in any execution of B. Additionally, if any transition t in any execution $\rho \in B$ is of the form $(q, q', \texttt{true}, \sigma, \tau)$, then transitions of the form $(q, q^*, \texttt{insample} < \texttt{x}, \sigma', \tau')$ or $(q, q^*, \texttt{insample} < \texttt{x}, \sigma', \tau')$ do not exist in any execution of B.

2. If a transition of the form $(q, q', \texttt{insample} < \texttt{x}, \sigma, \tau)$ exists in any execution in B and a transition of the form $(q, q^*, \texttt{insample} \geq \texttt{x}, \sigma', \tau')$ exists in any (potentially different) execution in B, then $\sigma \neq \sigma'$. Additionally, at least one of $\sigma \in \Gamma$, $\sigma' \in \Gamma$ is true.

We will refer to the first condition as **determinism** and the second condition as **output distinction**.

4.4.1 Privacy

Definition 4.25 (DP for branching programs)**.** A branching program is DP if [standard def]

Proposition 4.26. *A branching program B is $d\varepsilon$-differentially private for $d > 0$ if and only if for every complete execution ρ in B, ρ is $d\varepsilon$-differentially private.*

Proof. From the conditions of output distinction and determinism, note that for every single possible output σ of B, there is exactly one possible execution ρ that could have output σ. Thus, the probability that B outputs σ is exactly the probability that ρ outputs σ; the result follows immediately. □

We can do no better than assigning coupling strategies for each execution independently.

Definition 4.27 (Coupling strategies)**.** A (branched program) coupling strategy C for a branching program B is a collection of (execution) coupling strategies where each complete execution $\rho \in B$ is assigned a coupling strategy C_ρ.

Definition 4.28. A coupling strategy C for a branching program B is valid if, for every constituent execution coupling strategy C_ρ, C_ρ is valid.

Definition 4.29. The cost of a coupling strategy C for a branched program B is

$$\max_{\rho \in B} cost(\rho)$$

Proposition 4.30. *Optimal cost is dependent on execution (Vishnu)*

As previously noted, we use a finite union of complete executions to model an underlying program with branching, in part due to proposition 4.30. We now formally define that underlying model:

Definition 4.31. The underlying graph [NAME VERY TENTATIVE] of a branching program B is the directed acyclic graph $G = (V, E)$ such that $V = Q$ and $(q, q') \in E$ if, for some execution $\rho \in B$, there exists a transition $t \in \rho$ such that $t = (q, q', c, \sigma, \tau)$ for some c, σ, τ.

need also to force this to be acyclic

4.5 Loops

We will illustrate applying coupling strategies to loops through a simple **lasso** model.

Definition 4.32 (Lassos)**.** Consider a complete execution $\rho = q_0 \to \ldots \to q_n$ such that for some $i < n$, $q_i = q_n$. Suppose also that $\forall j \neq i$, $q_j \neq q_n$. A lasso L_ρ is the infinite set of executions $\{q_0 \to \ldots \to q_{i-1} \to (q_i \to \ldots \to q_{n-1})^k \to q_n | k \in \mathbb{N}\}$, where $(q_i \to \ldots \to q_{n-1})^k$ means that the subexecution $(q_i \to \ldots \to q_{n-1})^k$ is iterated k times. Additionally, L_ρ must satisfy the properties of **determinism** and **output distinction** as with branching programs.

Definition 4.33 (Lasso privacy)**.** A lasso L_ρ is $d\varepsilon$-differentially private for $d > 0$ if for all complete executions ρ' in L_ρ, for all possible outputs $\sigma_{\rho'}$ of ρ' and input sequences $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$, $\forall \varepsilon > 0, \mathbb{P}[\rho', \sigma_{\rho'}, \texttt{in}\langle 1\rangle] \leq e^{d\varepsilon}\mathbb{P}[\rho', \sigma_{\rho'}, \texttt{in}\langle 2\rangle]$.

Proposition 4.34. *A lasso L_ρ is $d\varepsilon$-differentially private for $d > 0$ if and only if for all complete executions ρ' in L_ρ, ρ' is $d\varepsilon$-differentially private.*

Definition 4.35 (Coupling strategies)**.** A **coupling strategy** for a lasso L_ρ, where $\rho = q_0 \to \ldots \to q_n$ is an assignment of shifts γ_i, γ_i' for each transition in ρ as a function of the input differences.

Definition 4.36 (Validity)**.** A coupling strategy $C_{L_\rho} = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ for a lasso L_ρ is **valid** if, for all executions ρ' of A, the following constraints hold for all adjacent input sequences $\alpha \sim \beta$ and for all i:

- If $c_i = \texttt{insample} < \texttt{x}$, then $\gamma_i \leq \gamma_{at(i)}$
- If $c_i = \texttt{insample} \geq \texttt{x}$, then $\gamma_i \geq \gamma_{at(i)}$
- If $\sigma_i = \texttt{insample}$, then $\gamma_i = 0$
- If $\sigma_i = \texttt{insample}'$, then $\gamma_i' = 0$

Note that given a complete execution ρ' in a lasso L_ρ, a coupling strategy C_A induces a coupling strategy $C_{\rho'}$ for ρ'.

Definition 4.37. The **cost** of a coupling strategy $C_{L_\rho} = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ for a lasso L_ρ is

$$\begin{aligned} cost(C_{L_\rho}) &= \sup_{\rho' \in L_\rho} \max_{\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle} \sum_{i=0}^{|\rho'|-1} (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i'|)d_i' \\ &= \sup_{\rho' \in L_\rho} cost(C_{\rho'}) \end{aligned}$$

Further, let G_{L_ρ} be the set of all valid coupling strategies for L_ρ. Then the **coupling cost** of L_ρ is

$$cost(L_\rho) = \min_{C_{L_\rho} \in G_{L_\rho}} cost(C_{L_\rho})$$

Proposition 4.38. *Given a valid coupling strategy C_{L_ρ} for a lasso L_ρ, L_ρ is $cost(C_{L_\rho})\varepsilon$-differentially private.*

Proof. Let ρ' be a complete execution in L_ρ and let $C_{\rho'}$ be the coupling strategy for ρ' derived from C_A. We know from lemma 4.17 that, if C_A is valid, then the lifting $\sigma\langle 1\rangle\{(a, b) : a = \sigma \implies b = \sigma\}^{\#cost(C_{\rho'})\varepsilon}\sigma\langle 2\rangle$ is valid, and therefore for all possible outputs σ of ρ' and adjacent inputs $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$, $\forall \varepsilon > 0, \mathbb{P}[\rho', \sigma, \texttt{in}\langle 1\rangle] \leq e^{cost(C_{\rho'})\varepsilon}\mathbb{P}[\rho', \sigma, \texttt{in}\langle 2\rangle] \leq e^{cost(C_{L_\rho})\varepsilon}\mathbb{P}[\rho', \sigma, \texttt{in}\langle 2\rangle]$. □

Corollary 4.39. *A lasso L_ρ is $cost(L_\rho)\varepsilon$-differentially private.*

Lemma 4.40. *For a lasso L_ρ, given adjacent input sequences $\boldsymbol{in}\langle 1\rangle \sim \boldsymbol{in}\langle 2\rangle$, a valid coupling strategy $C_{L_\rho} = (\gamma, \gamma')$ has finite cost $cost(C_{L_\rho}) < \infty$ if and only if the following constraint applies for all i:*

- *If t_i is in a cycle and $c_i \neq \texttt{true}$, then $\gamma_i = -\boldsymbol{in}\langle 1\rangle_i + \boldsymbol{in}\langle 2\rangle_i$ and $\gamma'_i = -\boldsymbol{in}\langle 1\rangle_i + \boldsymbol{in}\langle 2\rangle_i$.*

Proof. TODO: check the $c_i \neq \texttt{true}$ condition

($\Longleftarrow$)

Let T be the set of transitions t_i in L_ρ such that t_i is **not** in a cycle. Note that T is independent of any choice of execution(s) through L_ρ.

Fix a complete execution ρ' in L_ρ and let C'_ρ be the coupling strategy for ρ' induced by C_{L_ρ}.

Let D_ρ be the set of transitions t_i in ρ such that t_i is in a cycle in L_ρ, i.e., $t_i \notin T$.

If the given constraint holds, then we know that $\max_{\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle} \sum_{i:t_i \in D_\rho} (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma'_i|)d'_i = 0$

So

$$\begin{aligned}
cost(C_\rho) &= \max_{\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle} \sum_{i:t_i \in D_\rho} (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma'_i|)d'_i \\
&\qquad + \sum_{i:t_i \notin D_\rho} (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma'_i|)d'_i \\
&= \max_{\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle} \sum_{i:t_i \in T} (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma'_i|)d'_i \\
&\leq \sum_{i:t_i \in T} (2d_i + 2d'_i) \\
&\leq |T| \max_{i:t_i \in T} (2d_i + 2d'_i)
\end{aligned}$$

($\Longrightarrow$)

Suppose that some transition t_i is in a cycle C in A and $\gamma_i \neq -\texttt{in}\langle 1\rangle_i + \texttt{in}\langle 2\rangle_i$ or $\gamma'_i \neq -\texttt{in}\langle 1\rangle_i + \texttt{in}\langle 2\rangle_i$. Then $\exists \texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$ such that $(|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma'_i|)d'_i > 0$.

Fix such a $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$.

Let ρ_k be a complete execution in A with C iterated k times. Then for all $k \in \mathbb{N}$,

$$cost(\rho_k) \geq k((|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma'_i|)d'_i),$$

so for all $M \in \mathbb{R}$, $\exists \rho_k$ such that $cost(\rho_k) > M$. □

move these two lemmas away from this section (appendix or closer to usage in main theorem)

Lemma 4.41. *If a coupling strategy $C = (\gamma, \gamma')$ for a lasso L_ρ is valid and has finite cost, then the following must hold for all i:*

1. *If t_i is in a cycle and $c_i = \texttt{insample} < \texttt{x}$, then $\gamma_i = -\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle$ and $\gamma_{at(i)} = 1$.*
2. *If t_i is in a cycle and $c_i = \texttt{insample} \geq \texttt{x}$, then $\gamma_i = -\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle$ and $\gamma_{at(i)} = -1$.*

Proof. We will show (1). (2) follows symmetrically.

Consider some t_i in a cycle where $c_i = \texttt{insample} < \texttt{x}$. Because C is has finite cost, we know from lemma 4.40 that $\gamma_i = -\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle$ for all $\texttt{in}_i\langle 1\rangle \sim \texttt{in}_i\langle 2\rangle$. In particular, when $-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle = 1$, then $\gamma_i = 1$.

Further, because $\gamma_{at(i)}$ must be greater than γ_i for all $\texttt{in}_i\langle 1\rangle \sim \texttt{in}_i\langle 2\rangle$ for C to be valid, we must have that $\gamma_{at(i)} = 1$. □

Lemma 4.42. *If a valid finite cost coupling strategy $C = (\gamma, \gamma')$ exists for a lasso L_ρ, then there exists a valid finite cost coupling strategy $C^* = (\gamma^*, \gamma^{*\prime})$ such that for all $i \in AT(L_\rho)$, $\gamma_i^* \in \{-1, 0, 1\}$.*

Proof. TBD □

4.6 Programs

We will now define programs by adding loops to our basic branched program model:

Definition 4.43. A program A is the tuple $A = (Q, C, \Gamma, q_{init}, X, P, \delta)$ where

- Q is a finite set of states
- $C = \{\texttt{true}, \texttt{insample} < x, \texttt{insample} \geq x\}$ is a set of guard conditions
- Γ is a finite output alphabet
- $q_{init} \in Q$ is the initial state
- $X = \{\texttt{x}, \texttt{insample}, \texttt{insample}'\}$ is the set of variables
- $P : Q \to \mathbb{Q}^{\geq 0} \times \mathbb{Q}^{\geq 0}$ is a parameter function that assigns sampling parameters for the Laplace distribution for each state
- $\delta : (Q \times C) \to (Q \times (\Gamma \cup \{\texttt{insample}, \texttt{insample}'\}) \times \{\texttt{true}, \texttt{false}\})$ is a partial transition function.

Additionally, δ must satisfy the conditions:

- **Determinism:** For any state $q \in Q$, if $\delta(q, \texttt{true})$ is defined, then $\delta(q, \texttt{insample} < x)$ and $\delta(q, \texttt{insample} \geq x)$ are not defined.
- **Output Distinction:** For any state $q \in Q$, if $\delta(q, \texttt{insample} \geq x) = (q_1, o_1, b_1)$ and $\delta(q, \texttt{insample} < x) = (q_2, o_2, b_2)$, then $o_1 \neq o_2$ and at least one of $o_1 \in \Gamma$ and $o_2 \in \Gamma$ is true.
- **Initialization:** The initial state q_0 has only one outgoing transition of the form $\delta(q_0, \texttt{true}) = (q, o, \texttt{true})$.

Definition 4.44 (Program executions)**.** An execution of a program $A = (Q, C, \Gamma, q_{init}, X, P, \delta)$ is a sequence of transitions $\rho = t_0, \ldots t_{n-1}$ such that for all $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$, $\delta(q_i, c_i) = (q_{i+1}, \sigma_i, \tau_i)$.

Additionally, if $t_0 = (q_{init}, q_1, \texttt{true}, \sigma_0, \texttt{true})$ for some $q_1 \in Q, \sigma_0 \in \Gamma \cup \{\texttt{insample}, \texttt{insample}'\}$, then ρ is a complete execution.

Note that we are simply translating between languages of transitions and the finite state automata of programs. Every program A defines a possibly infinite set of its possible executions.

Definition 4.45 (DP)**.** Privacy Def

As with branching programs, the conditions of determinism and output distinction mean that knowing the output of a program means that we know what execution the program must have taken.

Proposition 4.46. *For every complete execution ρ of a program A, ρ is $d\varepsilon$-differentially private if and only if A is differentially private.*

As expected, then, we can use coupling strategies for executions to prove privacy.

Proposition 4.47. *If, for every complete execution ρ of a program A, there is a coupling strategy C_ρ such that $cost(C_\rho) < d$, then A is $d\varepsilon$-differentially private.*

However, this naive is not actually tractable, since there are possibly an infinite number of executions in a single program A due to cycles.

What we will show instead is that, inspired by our work with lassos, we can choose a coupling strategy for each of a finite number of subgraphs of A, which will induce coupling strategies for every execution of A.

Definition 4.48. For an execution ρ of a program A, let $graph(\rho)$ be the subgraph of the underlying graph of A that is traversed by ρ. That is, if the underlying graph of A is $G = (V, E)$, then $graph(\rho) = (V', E')$, where $V' \subseteq V$, $E' \subseteq E$, and for all transitions $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$, $q_i, q_{i+1} \in V$ and $(q_i, q_{i+1}) \in E$.

Note that the relation between complete executions $\sim_G$ where $\rho \sim_G \rho'$ if and only if $graph(\rho) = graph(\rho')$ is clearly an equivalence relationship; let $[\rho]_G$ be the equivalence class of ρ defined by this relationship for any complete execution ρ of a program A.

Definition 4.49 (Class Coupling Strategies)**.** Consider an execution class $[\rho]_G$ of a program A and let $G = (V, E) = graph(\rho)$ be the underlying subgraph of $[\rho]_G$. A [name tentative] class coupling strategy $C_{[\rho]_G}$ for $[\rho]_G$ is a function $C_{[\rho]_G} : E \to (\mathbb{R}\times\mathbb{R} \to [-1, 1])\times(\mathbb{R}\times\mathbb{R} \to [-1, 1])$ mapping edges of G to shift functions (γ, γ').

Observe that there are a finite possible number of execution classes of a program A since A must have a finite number of states; additionally, a class coupling strategy for an execution class $[\rho]_G$ directly induces execution coupling strategies for every complete execution in the class.

Definition 4.50. The cost of a class coupling strategy $C_{[\rho]_G}$ for an execution class $[\rho]_G$ is $cost(C_{[\rho]_G}) = \max_{\rho' \in [\rho]_G} cost(C_{\rho'})$, where $C_{\rho'}$ is the execution coupling strategy for ρ' induced by $C_{[\rho]_G}$.

Proposition 4.51. *If, for every complete execution class $[\rho]_G$ of a program A, there is a coupling strategy $C_{[\rho]_G}$ such that $cost(C_{[\rho]_G}) < d$, then A is $d\varepsilon$-differentially private.*

Indeed, for the purposes of privacy, we do not need to distinguish between different members of execution classes.

Proposition 4.52. *If there exists a valid and finite cost coupling strategy for every execution in a program A, then there exists a class coupling strategy for each execution class $[\rho]_G$ of A that is valid and has finite cost. also the same cost?*

Proposition 4.53. *If, for all execution classes $[\rho]_G$ of a program A, there exists a coupling strategy $C_{[\rho]_G}$ such that $cost(C_{[\rho]_G}) \leq d$, then A is $d\varepsilon$-differentially private.*

As before, for a coupling strategy to have finite cost, we must ensure that every cycle transition has no cost:

Proposition 4.54. *A valid class coupling strategy $C_{[\rho]_G}$ of A has finite cost if and only if the following constraint holds for every distinct transition t_i in $[\rho]_G$:*

- *If t_i is in a cycle in graph(ρ) and $c_i \neq$ `true`, then $\gamma_i = -\boldsymbol{in}\langle 1\rangle_i + \boldsymbol{in}\langle 2\rangle_i$ and $\gamma_i' = -\boldsymbol{in}\langle 1\rangle_i + \boldsymbol{in}\langle 2\rangle_i$.*

Surprisingly, we prove that

4.7 DiPA

Definition 4.55 (DiPA)**.** A DiPA A is a tuple (...) [copy this def]

Definition 4.56. A DiPA A is differentially private if [copy this def]

Proposition 4.57 (informal)**.** *The class of programs captured by [above] is exactly the set of programs captured by DiPA.*

4.7.1 Deciding Privacy

Lemma 4.58. *If there exists some valid coupling strategy S for a DiPA A with $cost(S) < \infty$, then A is $cost(S)$-DP.*

Proof. Follows immediately from [before]. □

Definition 4.59 (Leaking Pair, etc from DiPA)**.** [...]

Theorem 4.60 (from DiPA)**.** *If a DiPA A has a leaking cycle, leaking pair, disclosing cycle, or privacy violating path, then A is not $d\varepsilon$-differentially private for any $d > 0$.*

Note that for any path in A, we have a list of constraints that a coupling strategy $C = (\gamma, \gamma')$ can satisfy

1. If $c_i = \texttt{insample} < \texttt{x}$, then $\gamma_i \leq \gamma_{at(i)}$
2. If $c_i = \texttt{insample} \geq \texttt{x}$, then $\gamma_i \geq \gamma_{at(i)}$
3. If $\sigma_i = \texttt{insample}$, then $\gamma_i = 0$
4. If $\sigma_i = \texttt{insample}'$, then $\gamma'_i = 0$
5. If t_i is in a cycle and $c_i \neq \texttt{true}$, then $\gamma_i = -\texttt{in}\langle 1\rangle_i + \texttt{in}\langle 2\rangle_i$
6. If t_i is in a cycle and $c_i \neq \texttt{true}$, then $\gamma'_i = -\texttt{in}\langle 1\rangle_i + \texttt{in}\langle 2\rangle_i$

If all constraints are statisfied, then A is private.

Lemma 4.61. *If there does not exist a valid coupling strategy for a DiPA A with finite cost, then A is not differentially private.*

Proof. Consider a "maximially" satisfied coupling strategy $C = (\gamma, \gamma')$; i.e. there is no other coupling strategy C' for A such that C' satisfies more constraints than C. By lemma [tbd], we are allowed to only consider coupling strategies $C = (\gamma, \gamma')$ such that, for all $i \in AT(A)$, $\gamma_i \in \{-1, 0, 1\}$.

Fix some lasso ρ in A such that at least one constraint is not satisfied by C as applied to ρ.

By assumption, at least one constraint is unsatisfied by C. We will show that in every case, A must contain at least one of a leaking cycle, leaking pair, disclosing cycle, or privacy violating path. By theorem 4.60, this is sufficient to show that A is not $d\varepsilon$-differentially private for any $d > 0$.

Case 1: (1) is unsatisfied for γ_i

In this case, $c_i = \texttt{insample} < \texttt{x}$ and $\gamma_i > \gamma_{at(i)}$. Note that $\gamma_{at(i)} \neq 1$.

There are two ways to resolve constraint (1): either setting $\gamma_i = \gamma_{at(i)}$ or $\gamma_{at(i)} = 1$.

Because C is maximal, we know that setting $\gamma_i = \gamma_{at(i)}$ must violate some other constraint. In particular, it can either violate constraints (3) or (5) for γ_i or constraint (1) for some γ_j such that $at(j) = i$.

Similarly, if we set $\gamma_{at(i)} = 1$, then constraints (1), (3), or (5) for $\gamma_{at(i)}$ or constraint (2) for some γ_k such that $at(k) = at(i)$ would be violated.

Case 1a: Setting $\gamma_i = \gamma_{at(i)}$ would violate (3) for γ_i

In this case, then $\sigma_i = \texttt{insample}$. Additionally, $\gamma_{at(i)} \neq 0$, so $\gamma_{at(i)} = -1$.

Changing $\gamma_{at(i)}$ from -1 to 0 to satisfy contraint (3) for γ_i can newly violate constraints (1) or (5) for $\gamma_{at(i)}$ or constraint (2) for some γ_j such that $at(j) = at(i)$. We appeal to case (3a) to show that in such a scenario, there must be a leaking cycle, disclosing cycle, or privacy violating path.

Case 1b: Setting $\gamma_i = \gamma_{at(i)}$ would violate (5) for γ_i

The only way to satisfy both constraints (1) for γ_i and (5) is if $\gamma_{at(i)} = 1$. We then appeal to cases (1d-1g).

Case 1c: Setting $\gamma_i = \gamma_{at(i)}$ would violate (1) for some γ_j such that $at(j) = i$

In this case, γ_i was changed from 1 to $\gamma_{at(i)} \in \{-1, 0\}$; additionally, t_i is an assignment transition. We can assume that t_i is not in a cycle, since if it were, the cycle containing t_i would be a leaking cycle.

Note that $c_j = \texttt{insample} < \texttt{x}$.

If $\gamma_i = 0$, then $\gamma_j > 0$ originally. Then to satisfy constraint (1) for γ_j, we must either assign $\gamma_{at(i)} = \gamma_i = 1$ or assign $\gamma_j = 0$. If we assign $\gamma_{at(i)} = \gamma_i = 1$, then we appeal to cases (1d-1g). Otherwise, we will show that even if we assign $\gamma_j = 0$, we must still assign $\gamma_{at(i)} = \gamma_i = 1$ to attempt to satisfy more constraints. If we assign $\gamma_j = 0$, then either constraint (5) for γ_j or constraint (1) for some γ_k such that $at(k) = j$ must be newly violated. Note that (2) cannot be newly violated for some γk such that $at(k) = j$ since $j > 0$ originally; if (2) was satisfied when $j > 0$, then (2) must still be satisfied when $j = 0$.

If constraint (5) for γ_j is newly violated, then the only way to satisfy (1) for γ_j is to set $\gamma_{at(j)} = \gamma_i = \gamma_{at(i)} = 1$. As before, we appeal to cases (1d-1g).

If assigning $\gamma_j = 0$ would newly violate constraint (1) for some γ_k such that $at(k) = j$, then

???

come back to this

If $\gamma_i = -1$, then $\gamma_j > -1$. Satisfying (1) for γ_j by setting $\gamma_j = -1$ can possibly violate constraints (3) or (5) for

Case 1d: Setting $\gamma_{at(i)} = 1$ would violate (1) for $\gamma_{at(i)}$

Case 1e: Setting $\gamma_{at(i)} = 1$ would violate (3) for $\gamma_{at(i)}$

Case 1f: Setting $\gamma_{at(i)} = 1$ would violate (5) for $\gamma_{at(i)}$

Case 1g: Setting $\gamma_{at(i)} = 1$ would violate (2) for some γ_k such that $at(k) = at(i)$

Case 2: (2) is unsatisfied for γ_i

This case is exactly symmetric to case (1).

Case 3: (3) is unsatisfied for γ_i

First note that if t_i is in a cycle, then that cycle will be a disclosing cycle because t_i outputs `insample`. Thus, we will assume that t_i is not in a cycle.

Because C is maximal, setting $\gamma_i = 0$ must violate at least one of constraints (1) or (2) for γ_i or (1) for some γ_l such that $at(l) = i$.

Case 3a: Satisfying (3) for γ_i would violate (1) for γ_i

This means that $\gamma_{at(i)} < 0 \implies \gamma_{at(i)} = -1$. Further, $c_i = \texttt{insample} < \texttt{x}$. Then changing $\gamma_{at(i)} = 0$ can newly violate constraints (1) or (5) for $\gamma_{at(i)}$ or constraint (2) for some γ_j such that $at(j) = at(i)$.

If constraint (5) is newly violated, then $t_{at(i)}$ is in a cycle. In particular, the cycle must be a leaking cycle; if t_i and $t_{at(i)}$ are both contained in a cycle, then it must be leaking because $c_i = \texttt{insample} < \texttt{x}$. Otherwise, there still must be some transition in the cycle containing $t_{at(i)}$ that has a non-`true` guard since otherwise a path from $t_{at(i)}$ to t_i could not exist.

By similar reasoning, we can assume that for every assignment transition $t_{at(j)}$before $t_{at(i)}$ on a complete path to t_i, $t_{at(j)}$ is not in a cycle.

If constraint (1) is newly violated for $\gamma_{at(i)}$, then $c_{at(i)} = \texttt{insample} < \texttt{x}$. Let $t_{at(j)}$ be the earliest assignment transition before $t_{at(i)}$ such that $\gamma_{at(l)} = -1$ and for all assignment transitions $t_{at(k)}$ between $t_{at(j)}$ and $t_{at(i)}$, $c_{at(k)} = \texttt{insample} < \texttt{x}$ and $\gamma_{at(k)} = -1$.

Then there must exist some assignment transition $t_{at(k)}$, $at(j) \leq at(k) \leq at(i)$ between $t_{at(j)}$ and $t_{at(i)}$ such that setting $\gamma_{at(k)} = 0$ would newly violate constraint (2) for some l where $at(l) = at(k)$. In particular, this must be because t_l is in a cycle and setting $\gamma_l = 0$ would violate constraint (5). Thus, t_l is in a `G`-cycle. Then there is an `AL`-path from t_l to t_i, creating a privacy violating path from t_l to t_i.

If changing $\gamma_{at(i)}$ from -1 to 0 means that constraint (2) would be newly violated for some γ_j such that $at(j) = at(i)$, note that $\gamma_j < 0$ and $c_j = \texttt{insample} \geq \texttt{x}$.

So setting $\gamma_j = 0$ can violate either (2) for some γ_l where $at(l) = j$ or (5) for γ_j.

If setting $\gamma_j = 0$ would violate constraint (2) for some γ_l where $at(l) = j$, then let let $t_{at(m)}$ be the latest assignment transition after $t_{at(j)}$ such that $\gamma_{at(m)} = -1$ and for all assignment transitions $t_{at(k)}$ between $t_{at(j)}$ and $t_{at(m)}$, $c_{at(k)} = \texttt{insample} \geq \texttt{x}$ and $\gamma_{at(k)} = -1$.

Then there must exist some assignment transition $t_{at(k)}$, $at(j) \leq at(k) \leq at(m)$ between $t_{at(j)}$ and $t_{at(m)}$ such that setting $\gamma_{at(k)} = 0$ would newly violate constraint (2) for some l' where $at(l') = at(k)$. In particular, this must be because $t_{l'}$ is in a cycle and setting $\gamma_l = 0$ would violate constraint (5). Thus, t_l is in a `G`-cycle. Then there is an `AG`-path from t_i to t_l, creating a privacy violating path from t_i to t_l.

Otherwise, if setting $\gamma_j = 0$ would violate constraint (5) for γ_j, then t_j is in a `G`-cycle. We can assume that $j \neq i$ because otherwise, the cycle containing t_j would be a disclosing cycle. Additionally, note that there are no assignment transitions between t_i and t_j or vice versa, since $at(j) = at(i)$. Thus, if $j < i$, then there is an `AL`-path from t_j to t_i, which forms a privacy violating path. Symmetrically, if $i < j$, then there is an `AG`$-$path from t_i to t_j, which again forms a privacy violating path.

Case 3b: Satisfying (3) for γ_i would violate (2) for γ_i

This case is exactly symmetric to case (3a).

Case 3c: Satisfying (3) for γ_i would violate (1) for some γ_l where $at(l) = i$

Note that t_i must be an assignment transition. Further, we know that $\gamma_l > 0$ and $c_l = \texttt{insample} < \texttt{x}$.

Because C is maximal, setting $\gamma_l = 0$ would now violate either constraint (1) for some $\gamma_{l'}$ where $at(l') = l$ or constraint (5) for γ_l. Note that because $\gamma_l > 0$, constraint (2) cannot be newly violated for some $\gamma_{l'}$ where $at(l') = l$.

If constraint (5) would be newly violated for γ_l, then γ_l is in an L-cycle. Additionally, note that the path from t_{i+1} to t_l is an AL-path, so there is a privacy violating path from t_i to t_l.

Otherwise, if setting $\gamma_l = 0$ would violate constraint (1) for some $\gamma_{l'}$ where $at(l') = l$, let $t_{at(j)}$ be the latest assignment transition such that $c_{at(j)} = \texttt{insample} < \texttt{x}$ and $\gamma_{at(j)} < 1$ and, for all assignment transitions $t_{at(k)}$ between t_l and $t_{at(j)}$, $c_{at(k)} = \texttt{insample} < \texttt{x}$ and $\gamma_{at(k)} < 1$.

If $at(j) = l$, then l' is not an assignment transition. Then, setting $\gamma_{l'} = 0$ could only violate constraint (5). In this case, as before, there is a privacy violating path from t_i to t_l.

Otherwise, since C is maximal, we cannot set $\gamma_{at(k)} = 0$ for any $l < at(k) \leq at(j)$ without violating another constraint. In particular, there must be some $at(k)$ such that setting $\gamma_{at(k)} = 0$ would violate constraint (1) for some $\gamma_{k'}$ such that $at(k') = at(k)$. Note that there must be an AL-path from t_i to $t_{k'}$. Then, as before, there must be a privacy violating path from t_i to $t_{k'}$.

Case 3d: Satisfying (3) for γ_i would violate (2) for some γ_l where $at(l) = i$

This case is exactly symmetric to case (3c).

Case 4: (4) is unsatisfied for γ'_i

Because C is maximal, setting $\gamma'_i = 0$ must violate some other constraint. In particular, this must mean that constraint (6) is now violated. However, this would imply that t_i is in a cycle, and so the cycle containing t_i would be a disclosing cycle.

Case 5: (5) is unsatisfied for t_i: Because C is maximal, we know that if $\gamma_i = -\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle$ then another constraint must be violated. In particular, at least one of constraints (1), (2), or (3) must be violated for γ_i.

Case 5a: Satisfying (5) for t_i would violate (1)

If (1) is now violated, then either t_i is an assignment transition or $c_i = \texttt{insample} < \texttt{x}$ and $\gamma_{at(i)} < 1$. If t_i is an assignment transition, then the cycle containing t_i has a transition with a non-`true` guard (t_i) and an assignment transition, so it must be a leaking cycle.

Otherwise, if t_i is not an assignment transition, $c_i = \texttt{insample} < \texttt{x}$, and constraint (1) is violated for γ_i, we must have that $\gamma_{at(i)} < 1$ due to other constraints.

Consider all assignment transitions in ρ before t_i. Note that if any such assignment transition is in a cycle, then that cycle must be a leaking cycle since either the assignment transition is in the same cycle as t_i or there must be some non-`true` transition in the cycle because otherwise t_i is unreachable. make sure to add a condition to programs so that this is bad

So assume that all assignment transitions in ρ before t_i are not in a cycle. Then if $c_{at(i)} \neq \texttt{insample} < \texttt{x}$, because C is maximal, this must mean that $t_{at(i)}$ outputs $\texttt{insample}$. Note that the path from $t_{at(i)+1}$ to t_i is an $\texttt{AL}$-path (since there are no assignment transitions on it) and t_i is in an $\texttt{L}$-cycle since t_i is in a cycle and $c_i = \texttt{insample} < \texttt{x}$. Then the path from $t_{at(i)}$ (an assignment transition that outputs $\texttt{insample}$) to t_i is a privacy violating path.

If $c_{at(i)} = \texttt{insample} < \texttt{x}$, then let $c_{at(j)}$ be the earliest assignment transition such that $c_{at(j)} = \texttt{insample} < \texttt{x}$ and $\gamma_{at(j)} < 1$ and, for all assignment transitions $t_{at(k)}$ between $t_{at(j)}$ and t_i, $c_{at(k)} = \texttt{insample} < \texttt{x}$ and $\gamma_{at(k)} < 1$. Note that such an $t_{at(j)}$ must exist.

If $t_{at(j)} = t_{at(i)}$, then setting $\gamma_{at(i)} = 1$ must violate either constraint (2) for some other γ_l such that $at(l) = at(i)$, or constraint (3) for $\gamma_{at(i)}$. Without loss of generality, we will assume that $l \neq i$. If constraint (3) would be violated, then as before, there exists a privacy violating path from $t_{at(j)}$ to t_i. If constraint (2) would be violated for some γ_l such that $at(l) = at(i)$, then either t_l must output $\texttt{insample}$ or t_l must be in a cycle.

Suppose that $i < l$; then that the path from t_i to t_l is both an $\texttt{AG}$-path and an $\texttt{AL}$-path (since there are no assignment transitions on it). Thus, if t_l outputs $\texttt{insample}$, there exists a privacy violating path from t_i to t_l and if t_l is in a cycle, then the cycle containing t_i and the cycle containing t_l together make up a leaking pair, since the cycle containing t_l is a $\texttt{G}$-cycle by definition. Symmetrically, if $l > i$, then either the path from t_l to t_i is a privacy violating path or the cycle containing t_l and the cycle containing t_i make up a leaking pair.

Otherwise, note that the path from $t_{at(j)}$ to t_i is an $\texttt{AL}-$path. Since C is maximal, we cannot set $\gamma_{at(k)} = 1$ for $\gamma_{at(j)}$ or for any of the other assignment transitions $t_{at(k)}$ between $t_{at(j)}$ and t_i without violating another constraint. In particular, there must be some $t_{at(k)}$ where $at(j) \leq at(k) < i$ such that setting $\gamma_{at(k)} = 1$ would mean that either constraint (2) for some γ_l such that $at(l) = at(k)$ or constraint (3) would be violated for $\gamma_{at(k)}$. If constraint (3) would be violated for $\gamma_{at(k)}$ then $t_{at(k)}$ outputs $\texttt{insample}$, so as before, there is a privacy violating path from $t_{at(k)}$ to t_i. Otherwise if constraint (2) would be violated for some γ_l such that $at(l) = at(k)$, then as before, γ_l must either output $\texttt{insample}$ or t_l is in a cycle. Just like before, this means that there must be either a privacy violating path from t_l to t_i or the cycle containing t_l and the cycle containing t_i together make up a leaking pair.

Case 5b: Satisfying (5) for t_i would violate (2)

This case is exactly symmetric to case (5a).

Case 5c: Satisfying (5) for t_i would violate (3)

If (3) would be violated, then t_i must output $\texttt{insample}$. Then the cycle containing t_i must be a disclosing cycle.

Case 6: (6) is unsatisfied for t_i: Because C is maximal, we know that if $\gamma_i' = -\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle$ then another constraint must be violated for γ_i'. In particular, constraint (4) must be violated, since no other constraint involves γ_i'. Then t_i is a transition in a cycle that outputs $\texttt{insample}'$, so A has a disclosing cycle. □

Theorem 4.62. *A DiPA A is $d\varepsilon$-differentially private for some $d > 0$ if and only if there*

exists some valid coupling strategy S for A with $cost(S) < \infty$.

4.8 An algorithm for deciding privacy

Observe that the constraints imposed on valid coupling strategies for a complete path ρ only depend on the shifts associated with *assignment transitions* in ρ.

In particular, this lends itself to conceptualizing complete paths by splitting them up based on assignment transitions.

Definition 4.63. A **segment** of a (complete) path $\rho = q_0 \to \ldots \to q_n$ is a subpath $q_i \to q_{i+1} \to \ldots \to q_j$ of ρ such that $t_i \in A_\rho$; for all $i < k < j$, $t_k \notin A_\rho$; and either $j = n$ or $t_j \in A_\rho$.

In other words, a segment is a subpath of ρ between two consecutive assignment transitions (or between the last assignment transition and the end of the path). Splitting up a path into segments thus allows us to think about a single value of `x` at a time.

Proposition 4.64 (Vishnu's segment graph overview here)**.**

4.9 Minimizing a privacy budget

If we have a differentially private program, we'd also like to optimize its privacy cost. We can do so via couplings:

Proposition 4.65. *Introduce linear program here - maybe this is a definition?*

Proposition 4.66. *Introduce approximate version here*

Proposition 4.67. *Optimal linear program always has cost at worst approx version; there is indeed a separation between the optimal + approx*

Proposition 4.68. *Approximate version is an approximation of the optimal by a factor linear in the number of segments.*

Proposition 4.69. *Approximate version can be solved in polytime*

Conjecture 4.70 (IF I HAVE TIME LOOK INTO THIS)**.** Optimal coupling cost $\leq$ cost given by original DiPA analysis.

Conjecture 4.71. Optimal coupling cost = "true" optimal privacy cost

5 DiPA

We now introduce the final extension to our program model.

Definition 5.1 ([6])**.** A DiP[8] Automaton (DiPA) A is a 7-tuple $(Q, \Sigma, C, \Gamma, q_{init}, X, P, \delta)$ where

[8] **Di**fferentially **P**rivate

- Q is a finite set of states partitioned into input states Q_{in} and non-input states Q_{non}.
- $\Sigma = \mathbb{R}$ is the input alphabet
- $C = \{\texttt{true}, \texttt{insample} < x, \texttt{insample} \geq x\}$ is a set of guard conditions
- Γ is a finite output alphabet
- $q_{init} \in Q$ is the initial state
- $X = \{\texttt{x}, \texttt{insample}, \texttt{insample}'\}$ is the set of variables
- $P : Q \to \mathbb{Q} \times \mathbb{Q}^{\geq 0} \times \mathbb{Q} \times \mathbb{Q}^{\geq 0}$ is a parameter function that assigns sampling parameters for the Laplace distribution for each state
- $\delta : (Q \times C) \to (Q \times (\Gamma \cup \{\texttt{insample}, \texttt{insample}'\}) \times \{\texttt{true}, \texttt{false}\})$ is a partial transition function.

In addition, δ must satisfy some additional conditions:

- **Determinism:** For any state $q \in Q$, if $\delta(q, \texttt{true})$ is defined, then $\delta(q, \texttt{insample} < x)$ and $\delta(q, \texttt{insample} \geq x)$ are not defined.
- **Output Distinction:** For any state $q \in Q$, if $\delta(q, \texttt{insample} \geq x) = (q_1, o_1, b_1)$ and $\delta(q, \texttt{insample} < x) = (q_2, o_2, b_2)$, then $o_1 \neq o_2$ and at least one of $o_1 \in \Gamma$ and $o_2 \in \Gamma$ is true.
- **Initialization:** The initial state q_0 has only one outgoing transition of the form $\delta(q_0, \texttt{true}) = (q, o, \texttt{true})$.
- **Non-input transition:** From any $q \in Q_{non}$, if $\delta(q, c)$ is defined, then $c = \texttt{true}$.

A DiPA A functions almost identically to the previous automata we have defined, except that instead of only the first transition of A assigning into $\texttt{x}$, now any **assignment transition** can re-assign the value of $\texttt{x}$ to be $\texttt{insample}$. In addition, DiPAs are no longer dependent on an initial value for $\texttt{x}$, since the first transition must always be a $\texttt{true}$ guard assignment transition.

Note that we can treat a DiPA A as a (possibly infinite, if an assignment transition is in a cycle) set of segment automata by "splitting" at every assignment transition in A.

Definition 5.2. For a path ρ in a DiPA A, let $segments(\rho)$ be the set of subpaths of ρ created by splitting $\rho = r_0 \to \ldots \to r_n$ at each assignment transition.

More precisely, let I be the ordered set of indices such that $\texttt{trans}(r_{I_i})$ for $i \in [|I|]$ is an assignment transition. Then $segments(\rho) = \{r_0 \to \ldots \to r_{I_0}, \ldots, r_0 \to \ldots \to r_{I_k}\}$.

Definition 5.3. Let A be a DiPA and let P be the set of all paths through A.

Then $branches(A) = \bigcup_{\rho \in P} \bigcup_{\eta \in segments(\rho)} acyclic(\eta)$ be the set of branches in A.

Note that every path in A can be broken up into a sequence of branches, each from its own segment.

5.1 Joining Coupling Strategies

Naively, the fact that we can separate a DiPA A into a set of segment automata would suggest an approach of assigning a coupling strategy to each branch of A as before and simply combining these coupling strategies in sequence. However, it is not always possible to arbitrarily join coupling strategies together.

This is primarily because we can no longer freely decide the initial value for `x` at the beginning of each segment, as we could with segment automata. Recall that for a fixed path $\rho = r_0 \to \ldots \to r_n$, it was important that the initial transition guard $\texttt{guard}(r_0)$ is satisfied. Notably, whether or not the initial guard is satisfied is dependent on the initialized value of `x`; since we had no additional restrictions on `x`, we were able to (for example) arbitrarily couple $x_0\langle 1\rangle + 1 (=)^{\#d_x\varepsilon} x_0\langle 2\rangle$.

However, now that we have joined multiple segment automata together, the initial value x_0 for a segment automata is actually the assigned value `x` from the *previous* segment automata. This means that $x_0\langle 1\rangle$ and $x_0\langle 2\rangle$ will already be coupled together, perhaps in a way such that coupling $\texttt{insample}_0\langle 1\rangle$ and $\texttt{insample}_0\langle 2\rangle$ will be impossible to both ensure the correct shift of `x` for the new coupling strategy and so that the initial guard is satisfied.

For example, consider two paths $\rho = r_0 \to \ldots \to r_n$ and $\rho' = q_1 \to \ldots \to q_m$ with the assignment transition $r_n \to q_1$ connecting them, where $\texttt{guard}(r_n) = \texttt{insample} \geq x$. We are aiming to choose one coupling strategy for each of ρ and ρ'.

If S^L is chosen for ρ, then at the connecting assignment transition, x_0 is coupled such that $x_0\langle 1\rangle + 1 = x_1\langle 1\rangle$. Further, we would like to couple $\texttt{insample}_0$ such that $\texttt{insample}_0\langle 1\rangle = \texttt{insample}_0\langle 2\rangle + 1$. However, trying to create this lifting while also ensuring that the implication $\texttt{insample}_0\langle 1\rangle \geq x_0\langle 1\rangle \implies \texttt{insample}_0\langle 2\rangle \geq x_0\langle 2\rangle$ holds is impossible.

Thus, we cannot choose S^G for a segment that follows a segment with coupling strategy S^L if the connecting assignment transition has guard $\texttt{insample} \geq x$. There are a series of similar restrictions on how coupling strategies can be connected for adjacent segments, which along with previous restrictions on coupling strategies (e.g. we cannot choose S^G or S^L if the first transition of a segment outputs `insample`).

5.2 A Constraint System for Valid Couplings

Based on the restrictions on joining segments together, we can construct a set of constraints on coupling strategies for a DiPA A that, if solved, will give us a valid proof of $d\varepsilon$-differential privacy.

Construction 5.4. Consider a DiPA A.

For each branch $s_i \in \mathit{branches}(A)$, we can assign one of three coupling strategies $S_i \in \{S^L, S^G, S^N\}$. We would like to find an assignment of coupling strategies for each segment of each variable, subject to the following constraints:

1. Constraints for valid couplings
   (a) For all s_i, if $\texttt{trans}(s_i)$ outputs `insample`, then $S_i = S^N$.

(b) For all s_i, s_j such that s_i is directly followed by s_j,

i. If $\texttt{guard}(s_j) = \texttt{insample} < x$ and $S_i = S^G$, then $S_j = S^G$.

ii. If $\texttt{guard}(s_j) = \texttt{insample} \geq x$ and $S_i = S^L$, then $S_j = S^L$.

iii. If $\texttt{guard}(s_j) = \texttt{insample} < x$ and $S_i = S^N$, then $S_j \neq S^L$.

iv. If $\texttt{guard}(s_j) = \texttt{insample} \geq x$ and $S_i = S^N$, then $S_j \neq S^G$.

2. Constraints for finite cost

(a) For all s_i, no cycle in s_i has a transition that outputs $\texttt{insample}$ or $\texttt{insample}'$.

(b) For all s_i, if s_i has an $\texttt{L}$-cycle, then $S_i = S^L$.

(c) For all s_i, if s_i has a $\texttt{G}$-cycle, then $S_i = S^G$.

(d) For all segments s_i, there is no transition $\texttt{trans}(a_k)$ in s_i that is *faulty*, i.e.:

i. If s_i contains a $\texttt{insample} < x$ transition that outputs $\texttt{insample}$, then $S_i \neq S^G$.

ii. If s_i contains a $\texttt{insample} \geq x$ transition that outputs $\texttt{insample}$, then $S_i \neq S^L$.

Theorem 5.5. *Let A be a DiPA. If $|branches(A)| < \infty$ and for each branch $B \in branches(A)$, there exists an assignment of coupling strategies to each segment that satisfies the constraint system defined in 5.4, then A is $d\varepsilon$-differentially private for*

$$d = \sum_{s_i \in branches(A)} d_{s_i},$$

where d_{s_i} is the cost of the satisfying assignment strategy S_i for a branch s_i.

Proof. Let ρ be a path through A that is composed of branches $s_{x_1} \to \ldots \to s_{x_k}$ and let $S_{x_1}, \ldots, S_{x_k}$ be the associated coupling strategies from the satisfying assignment.

Because of constraints (1a-1b), we can actually compose each coupling strategy in sequence to create a valid lifting $A(X)\{(a, b) : a = \sigma \implies b = \sigma\}A(X')$ for all adjacent datasets $X \sim X'$ and possible outputs σ of ρ.

In addition, we claim that the cost of the satisfying assignments d_{s_i} is bounded for all s_i. There are two ways for a branch cost d_{s_i} to be unbounded: either a cycle transition has non-zero cost, or a transition has to be faulty with respect to the chosen coupling strategy. From constraint (2d), we know that no transitions can be faulty, and we know from constraints (2a-2c) that every cycle transition has to have zero cost.

Finally, the fact that $|branches(A)| < \infty$ and every path can go through each branch at most one time gives us the bound on d. □

5.3 Well-formedness

Similar to before, [6] define four structures in the graphs of DiPAs that characterize privacy for DiPAs. Within DiPAs, these graph structures can be defined as follows:

- **Leaking Cycles**: A cycle C is a leaking cycle if one of the transitions in C is an assignment transition.
- **Leaking Pair**: A pair of cycles C, C' are a leaking pair if C is an `L`-cycle, C' is a `G`-cycle, and there exists a path from C to C' such that every assignment transition on the path has guard $\texttt{insample} \geq x$; or, symmetrically, C is an `G`-cycle, C' is a `L`-cycle, and there exists a path from C to C' such that every assignment transition on the path has guard $\texttt{insample} < x$.
- **Disclosing Cycle**: A cycle C is a disclosing cycle if a transition in C outputs either `insample` or $\texttt{insample}'$.
- **Privacy Violating Path**: A path $\rho = q_0 \to \ldots \to q_n$ is a privacy violating path if any of the following conditions hold:
  - $q_1 \to \ldots q_n$ is a path whose assignment transitions all have guard $\texttt{insample} \geq x$ (resp. $\texttt{insample} < x$) such that q_n is part of a `G`-cycle (resp. `L`-cycle) and $\texttt{trans}(q_0)$ is an assignment transition that outputs `insample`.
  - ρ is a path whose assignment transitions all have guard $\texttt{insample} \geq x$ (resp. $\texttt{insample} < x$) such that q_n is part of a `G`-cycle (resp. `L`-cycle), $\texttt{guard}(q_0) = \texttt{insample} < x$ (resp. $\texttt{insample} \geq x$), and $\texttt{trans}(q_0)$ outputs `insample`.
  - ρ is a path whose assignment transitions all have guard $\texttt{insample} \geq x$ (resp. $\texttt{insample} < x$) such that q_0 is part of a `L`-cycle (resp. `G`-cycle), $\texttt{guard}(q_{n-1}) = \texttt{insample} \geq x$ (resp. $\texttt{insample} < x$), and $\texttt{trans}(q_{n-1})$ outputs `insample`.

Definition 5.6. A DiPA A is **well-formed** if it does not have a leaking cycle, leaking pair, disclosing cycle, or privacy violating path.

Importantly, [6] have shown that the existence of these graph structures completely decides whether or not a DiPA is $d\varepsilon$-differentially private.

Theorem 5.7 ([6])**.** *A DiPA A is $d\varepsilon$-differentially private for some $d > 0$ if and only if A is well-formed. Further, the well-formedness of an automaton A can be decided in linear time in the size of A.*

We will leverage this result to show that the existence of valid coupling strategies is also a complete characterization of DiPAs.

Lemma 5.8. *For a DiPA A, $|branches(A)| < \infty$ if and only if A has no leaking cycles.*

Lemma 5.9. *Suppose a DiPA A has a finite number of branches. If constraint system 5.4 is not satisfiable, there must exist a leaking cycle, a leaking pair, a disclosing cycle, or a privacy violating path in the graph of A.*

Proof. TBD, want to briefly discuss the best way to do this.

high level overview: very similar to the previous, one-segment, version of this, but we can chain constraints across `AG−` and `AL−`paths.

The reasoning gets a bit convoluted though, because there are a bunch of cases / you have to keep a bunch of assumptions hanging around. □

Theorem 5.10. *A DiPA A is $d\varepsilon$-differentially private if and only if A has a finite number of branches and there exists a valid assignment of coupling strategies over all branches of A.*

Proof. Follows from theorem 5.5 and lemmas 5.8 and 5.9. □

6 Bounds on Privacy

So far, we have focused on the binary question of whether a DiPA is private or not for *any* finite $d > 0$. An obvious question remaining thus is how tight of a privacy cost bound can we obtain using couplings?

Recall that S^G, S^L, and S^N are primarily characterized by **shifts**; that is, offseting variables from each other.

Indeed, we can generalize our discrete set of coupling strategies to a continuous family of coupling strategies by treating the offsets of each variable as parameters. By rephrasing in this way, we can derive a linear system:

[insert vishnu's linear program here]

If solvable, this linear system will thus give the best possible privacy cost obtainable using this family of coupling strategies.

Proposition 6.1. *An optimal solution to the linear program provides the minimal cost coupling over all shift coupling strategies. If the linear program is infeasible, then the DiPA is not $d\varepsilon$-differentially private for any ε.*

Theorem 6.2 (optimistically)**.** *If a DiPA A is ε-differentially private, then there exists some valid assignment of coupling strategies over all segments that has a total cost of ε (i.e. coupling cost is tight).*

7 Multivariable DiPA

In this section, we explore a natural extension of the DiPA model to demonstrate the utility of using couplings as proof infrastructure. Specifically, we introduce Generalized DiPAs (GDiPAs), which allow for an arbitrary finite set of variables and an expanded alphabet where multiple variables can be compared to the input simultaneously to determine transitions. This would, for example, allow for SVT-style algorithms that check for membership between or outside of two thresholds or indeed, membership within any finite union of intersections of halves[9] of $\mathbb{R}$.

[9] think there's a term for this I'm forgetting

7.1 GDiPA

A GDiPA A is a generalization of DiPA whose primary characteristics are that:

- At every state, a (real-valued) input is read. Additionally, Laplace noise (with parameters set by the user) is added to the input.
- The automaton has a stored finite set of (real-valued) variables $\mathcal{X}$. At each transition, the value of each variable can be updated with the noisy input value read in.
- A transition can optionally assign the (noisy) input value read at the previous state into at most one program variable.
- The automaton will take transitions based on a boolean combination of comparisons between the noisy input and a subset of the stored variables. Importantly, noise is added independently to the input for each separate variable comparison. Alternatively, there can be exactly one guaranteed (`true`) transition out of a state.
- At every transition, the automaton will output either
  1. A symbol from a pre-defined finite alphabet (Γ)
  2. The noisy input value as compared to one of the input variables (`insample`)
  3. The input value with fresh Laplace noise (`insample`$'$)

  As with DiPAs, the output sequence of any GDiPA must uniquely determine a path through the automaton.

7.2 Combining Separate Variable Couplings

Consider a GDiPA A with program variables $\mathcal{X}$. For each program variable $x \in \mathcal{X}$, we can define coupling strategies similar to with single variables. That is, for two runs of the automaton with adjacent inputs, we must ensure that if the first run takes a certain transition, the other run does as well. In particular, for real outputs, we must ensure that the outputs are equal.

For each variable in A, we can consider a "shadow" automaton in a single variable that only has a single program variable. Such a shadow automaton would have the same underlying graph structure as A, but would only contain the transition guards and assignments that pertained to a single variable. For example, if we were considering such an automaton A_x with respect to the variable x, a transition with guard "$\texttt{insample} < x$ and $\texttt{insample} \geq y$" would correspond to a transition with guard $\texttt{insample} < x$ in A_x; a transition with guard "$\texttt{insample} \geq y$" would correspond to a transition with guard `true` in A_x.

Note that these automata are not, strictly speaking, DiPAs, since it is possible for a state to have multiple transitions with guard `true` leaving it.

Thus, each segment can be assigned a coupling strategy S^N, S^L, or S^G as before based on these "shadow" automata, with similar constraints.

Construction 7.1. Consider a GDiPA A with program variables $\mathcal{X}$. Let $\{s_i^{(x)}\}$ be the segments of A for each variable $x \in \mathcal{X}$.

For each $x \in \mathcal{X}$ and segment $s_i^{(x)}$ for x, we can assign one of three coupling strategies $S_i^{(x)} \in \{S^L, S^G, S^N\}$. We would like to find an assignment of coupling strategies for each segment of each variable, subject to the following constraints:

1. Constraints for valid couplings
   (a) For all $s_i^{(x)}$, if $\texttt{trans}(s_i^{(x)})$ outputs $\texttt{insample}$, then $S_i^{(x)} = S^N$.
   (b) For all $s_i^{(x)}, s_j^{(x)}$ such that $s_i^{(x)}$ is immediately followed by $s_j^{(x)}$,
      i. If $\texttt{guard}(s_j^{(x)}) = \texttt{insample} < x$ and $S_i^{(x)} = S^G$, then $S_j^{(x)} = S^G$.
      ii. If $\texttt{guard}(s_j^{(x)}) = \texttt{insample} \geq x$ and $S_i^{(x)} = S^L$, then $S_j^{(x)} = S^L$.
      iii. If $\texttt{guard}(s_j^{(x)}) = \texttt{insample} < x$ and $S_i^{(x)} = S^N$, then $S_j^{(x)} \neq S^L$.
      iv. If $\texttt{guard}(s_j^{(x)}) = \texttt{insample} \geq x$ and $S_i^{(x)} = S^N$, then $S_j^{(x)} \neq S^G$.
   (c) For all segments $s_i^{(x)}$, there is no transition $\texttt{trans}(a_k)$ in $s_i^{(x)}$ that is *faulty*, i.e.:
      i. If $s_i^{(x)}$ contains a $\texttt{insample} < x$ transition that outputs $\texttt{insample}$, then $S_i^{(x)} \neq S^G$.
      ii. If $s_i^{(x)}$ contains a $\texttt{insample} \geq x$ transition that outputs $\texttt{insample}$, then $S_i^{(x)} \neq S^L$.
   (d) If any transition in $s_i^{(x)}$ outputs x, then $S_i^{(x)} = S^N$.
2. Constraints for finite cost
   (a) For all $s_i^{(x)}$, no cycle in $s_i^{(x)}$ has a transition that outputs $\texttt{insample}$, $\texttt{insample}'$.
   (b) For all $s_i^{(x)}$, if $s_i^{(x)}$ has an $\texttt{L}$-cycle with respect to x, then $S_i^{(x)} = S^L$.
   (c) For all $s_i^{(x)}$, if $s_i^{(x)}$ has a $\texttt{G}$-cycle with respect to x, then $S_i^{(x)} = S^G$.

This constraint system is **satisfiable** for a GDiPA A if, for all variables $x \in \mathcal{X}$, there exists an assignment of all $S_i^{(x)}$ such that all constraints are satisfied.

In particular, we can independently resolve constraints for each variable and combine them together.

Theorem 7.2. *For a GDiPA $\mathcal{A}$, for all variables x, if there exist a finite number of segments s_i, and for all segments s_i, the constraint system 5.4 is satisfied by an assignment S_x, then all S_x assignments together induce a valid coupling proof that $\mathcal{A}$ is ε-DP.*

Proof. Fix some variable $x \in \mathcal{X}$. From before, we know that if 7.1 is satisfied with respect to x, then the coupling

$$\mathcal{A}(X)(\mathcal{A}(X) \text{ takes transitions } T \text{ wrt } x \implies \mathcal{A}(X') \text{ takes transitions } T \text{ wrt } x)^{\#(\varepsilon_x, 0)}\mathcal{A}(X')$$

is valid for some finite ε_x.

Because the noises on `insample` are independent, and $((a \implies c) \wedge (b \implies d)) \implies ((a \wedge b) \implies (c \wedge d))$ and $((a \implies c) \vee (b \implies d)) \implies ((a \vee b) \implies (c \vee d))$,for every combined guard, we can construct the lifting

$$\mathcal{A}(X)(\mathcal{A}(X) \text{ takes transitions } T \implies \mathcal{A}(X') \text{ takes transitions } T)^{\#(\sum_{x \in \mathcal{X}} \varepsilon_x, 0)} \mathcal{A}(X')$$

which gives us the lifting

$$\mathcal{A}(X)(\mathcal{A}(X) = \sigma \implies \mathcal{A}(X') = \sigma)^{\#(\sum_{x \in \mathcal{X}} \varepsilon_x, 0)} \mathcal{A}(X')$$

□

Theorem 7.3 (less optimistically)**.** *If, for each variable x, there exists a valid assignment of coupling strategies over all segments of a DiPA A with respect to x, then A is $d\varepsilon$-differentially private.*

Theorem 7.4 (optimistically)**.** *A GDiPA A is $d\varepsilon$-differentially private if and only if, for each variable x, there exists a valid assignment of coupling strategies over all segments of A with respect to x.*

8 Conclusion

We have shown how to use coupling techniques to prove privacy for a class of SVT-like programs first defined in [6] and discovered that couplings additionally characterize this class. We additionally showed that this can be done tractably, and that couplings can help provide lower bounds on privacy costs of these algorithms.

Future work most naturally would focus on extensions of the program model. For the model, potential areas include removing the requirement for output to be deterministic of a path through the automaton, which would allow for algorithms such as Report Noisy Max to be captured by the model. Similarly, the alphabet of the automaton could be expanded to incorporate more than comparisons between two real numbers. Such extensions would naturally also require extensions of the class of couplings we define here, which are limited to "shifts".

Additionally, we believe that couplings should completely characterize GDiPAs as well as DiPAs; proving this requires showing that a lack of well-formedness in any single variable generates a counterexample to privacy. In this vein, we would like to explore using couplings to *disprove* privacy; the fact that shift couplings completely characterize DiPAs hints at the possibility of "anti-couplings" to generate counterexamples.

9 Related Work

The DiPA model and counterexamples to privacy are drawn from [6]. Approximate liftings were developed in [5, 4] and applied to algorithms such as SVT in [3]. A full exploration of

approximate liftings can be found in [9]. [1] uses couplings; and in particular the "shift" couplings family we use, to create a heuristiccally successful program for proving the correctness of possible differentially private algorithms.

need to reformat some citations at some point

References

[1] Aws Albarghouthi and Justin Hsu. Synthesizing coupling proofs of differential privacy. 58:30, 2018.

[2] Gilles Barthe, Rohit Chadha, Vishal Jagannath, A. Prasad Sistla, and Mahesh Viswanathan. Deciding differential privacy for programs with finite inputs and outputs. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, page 141–154, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. *Proceedings - Symposium on Logic in Computer Science*, 05-08-July-2016:749–758, 1 2016.

[4] Gilles Barthe and Federico Olmedo. Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7966 LNCS:49–60, 2013.

[5] Gilles Barthe, Boris K ¨ Opf, Federico Olmedo, and Santiago Zanella-B ´ Eguelin. N probabilistic relational reasoning for differential privacy.

[6] Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan. On Linear Time Decidability of Differential Privacy for Programs with Unbounded Inputs, April 2021.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[8] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, aug 2014.

[9] Justin Hsu. Probabilistic couplings for probabilistic reasoning. *CoRR*, abs/1710.09951, 2017.

[10] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *Proc. VLDB Endow.*, 10(6):637–648, feb 2017.

[11] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and XiaoFeng Wang. Privacy loss in apple's implementation of differential privacy on macos 10.12. *CoRR*, abs/1709.02753, 2017.